Lecture 03 – Background + intro AI + dataflow and connection to AI

PROGRAM ANALYSIS & SYNTHESIS

Eran Yahav

Previously…

- structural operational semantics
- trace semantics
  - meaning of a program as the set of its traces
- abstract interpretation
  - abstractions of the trace semantics

Example: Cormac's PLDI'08

Example: Kuperstein PLDI'11

Today

- abstract interpretation
- some basic math background
  - lattices
  - functions
- Galois connection
- A little more dataflow analysis
  - monotone framework

Trace Semantics

$[y := x]^1;$
$[z := 1]^2;$
$\text{while } [y > 0]^3 \ ($
$\quad [z := z * y]^4;$
$\quad [y := y - 1]^5;$
$)$
$[y := 0]^6$

$$< 1,\{ x\mapsto 42, y\mapsto 0, z\mapsto 0 \} > \overset{[y := x]^1}{\Rightarrow} < 2,\{ x\mapsto 42, y\mapsto 42, z\mapsto 0 \} > \overset{[z := 1]^2}{\Rightarrow}$$

$$< 3,\{ x\mapsto 42, y\mapsto 42, z\mapsto 1 \} > \overset{[y > 0]^3}{\Rightarrow} < 4,\{ x\mapsto 42, y\mapsto 42, z\mapsto 1 \}> \ldots$$

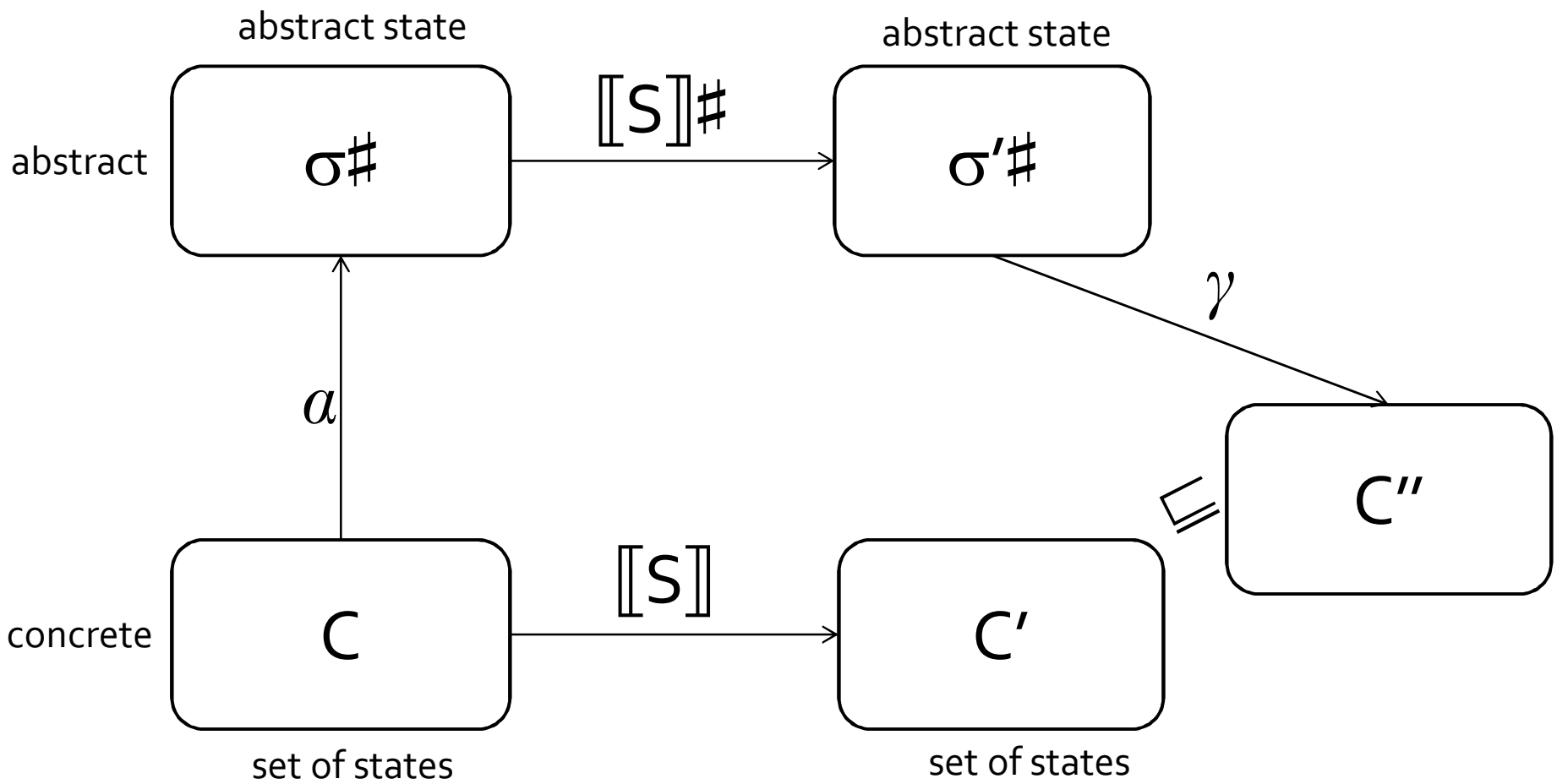
$$< 1,\{ x\mapsto 73, y\mapsto 0, z\mapsto 0 \} > \overset{[y := x]^1}{\Rightarrow} < 2,\{ x\mapsto 73, y\mapsto 73, z\mapsto 0 \} > \overset{[z := 1]^2}{\Rightarrow}$$

$$< 3,\{ x\mapsto 73, y\mapsto 73, z\mapsto 1 \} > \overset{[y > 0]^3}{\Rightarrow} < 4,\{ x\mapsto 73, y\mapsto 73, z\mapsto 1 \}> \ldots$$

$$\vdots$$

- note that input (x) is unknown
- while loop?
- trace semantics is not computable

Abstract Interpretation



Dataflow Analysis

```
                    { (x,?), (y,?), (z,?) }
1: y := x;
                    { (x,?), (y,1), (z,?) }
2: z := 1;
                    { (x,?), (y,1), (z,2) }
3: while y > 0 {
                    { (x,?), (y,1), (z,2) }
4:    z := z * y;
                    { (x,?), (y,?), (z,4) }
5:    y := y - 1
                    { (x,?), (y,5), (z,4) }
 }
                    { (x,?), (y,1), (z,2) }
6: y := 0
                    { (x,?), (y,?), (z,?) }
```

- Reaching Definitions
  - The assignment **lab: var := exp** reaches **lab'** if there is an execution where **var** was last assigned at **lab**

(adapted from Nielson, Nielson & Hankin)

Dataflow Analysis

```
                    { (x,?), (y,?), (z,?) }
1: y := x;
                    { (x,?), (y,1), (z,?) }
2: z := 1;
                    { (x,?), (y,1), (z,2), (y,5), (z,4) }
3: while y > 0 {
                    { (x,?), (y,1), (z,2), (y,5), (z,4) }
4:    z := z * y;
                    { (x,?), (y,?), (z,4), (y,5) }
5:    y := y – 1
                    { (x,?), (y,5), (z,4) }
 }
                    { (x,?), (y,1), (z,2), (y,5), (z,4) }
6: y := 0
                    { (x,?), (y,6), (z,2), (z,4) }
```

- Reaching Definitions
  - The assignment **lab: var := exp** reaches **lab'** if there is an execution where **var** was last assigned at **lab**

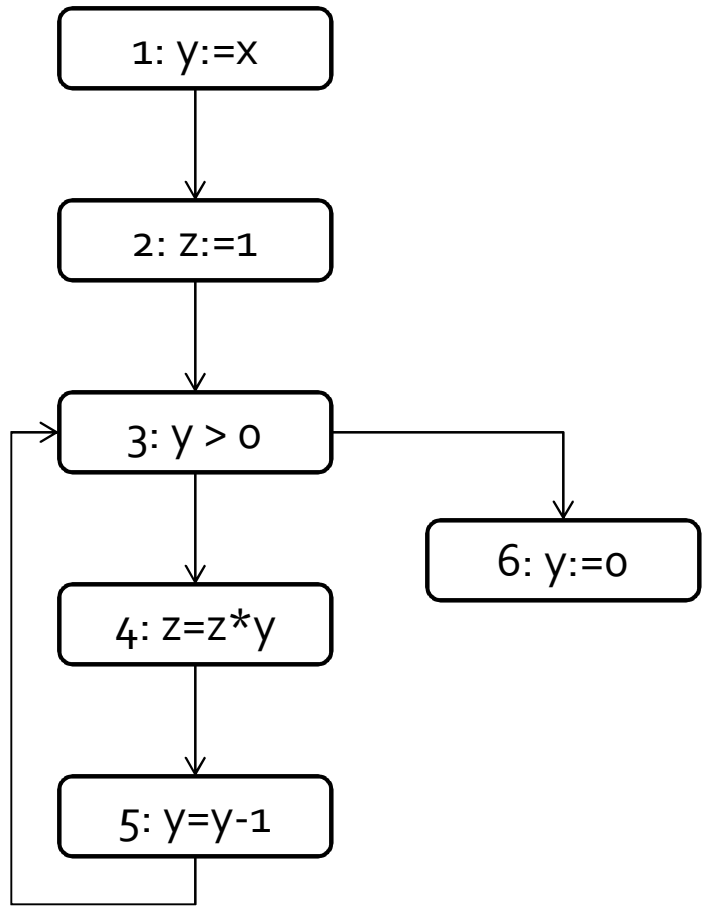
(adapted from Nielson, Nielson & Hankin)

Dataflow Analysis

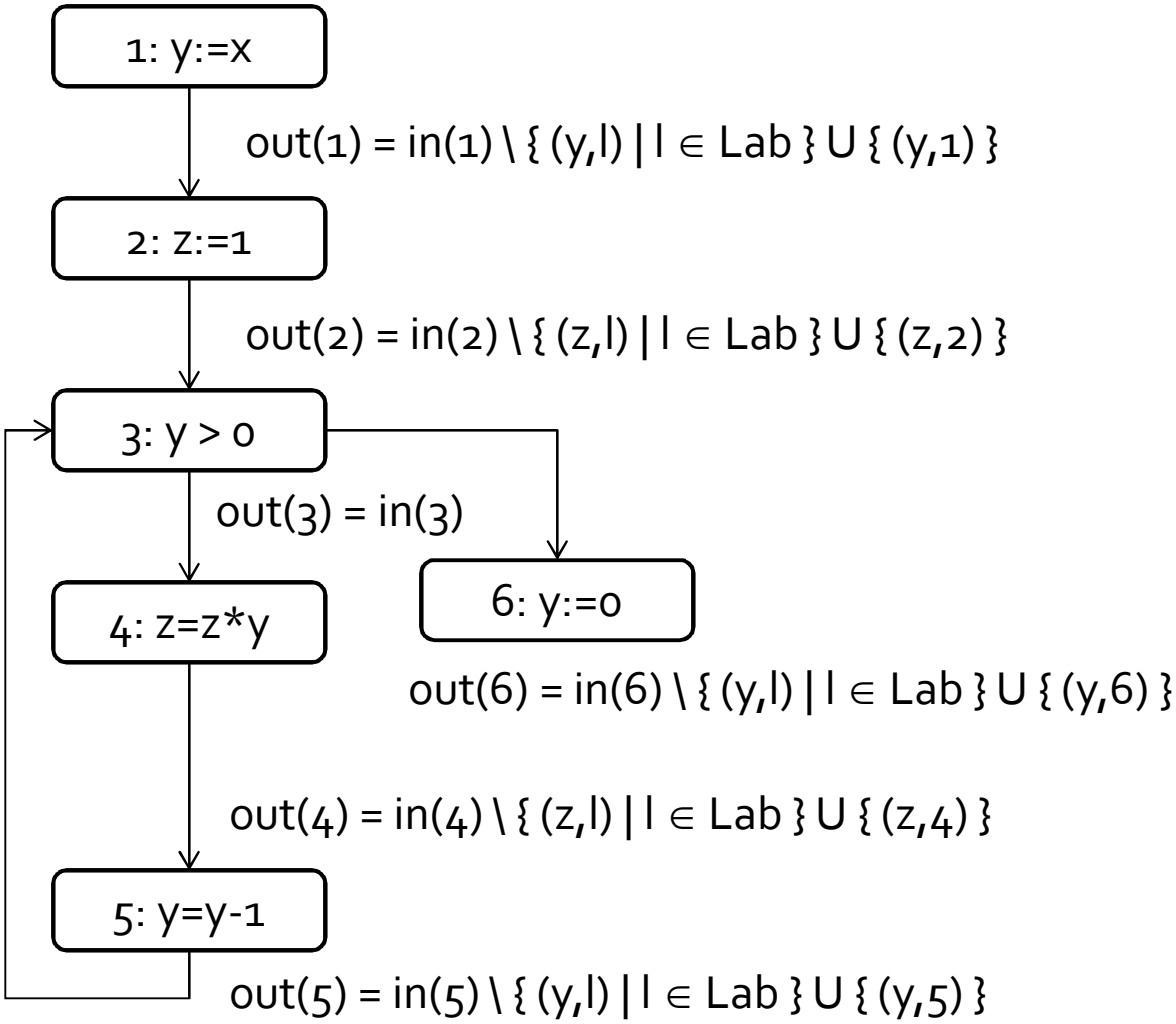
- Build control-flow graph
- Assign transfer functions
- Compute fixed point

Control-Flow Graph

```
1: y := x;
2: z := 1;
3: while y > 0 {
4:    z := z * y;
5:    y := y – 1
 }
6: y := 0
```



Transfer Functions



in(1) = { (x,?), (y,?), (z,?) }
in(2) = out(1)
in(3) = out(2) U out(5)
in(4) = out(3)
in(5) = out(4)
in(6) = out(3)

System of Equations

$in(1) = \{ (x,?), (y,?), (z,?) \}$
$in(2) = out(1)$
$in(3) = out(2) \cup out(5)$
$in(4) = out(3)$
$in(5) = out(4)$
$In(6) = out(3)$
$out(1) = in(1) \setminus \{ (y,l) \mid l \in Lab \} \cup \{ (y,1) \}$
$out(2) = in(2) \setminus \{ (z,l) \mid l \in Lab \} \cup \{ (z,2) \}$
$out(3) = in(3)$
$out(4) = in(4) \setminus \{ (z,l) \mid l \in Lab \} \cup \{ (z,4) \}$
$out(5) = in(5) \setminus \{ (y,l) \mid l \in Lab \} \cup \{ (y,5) \}$
$out(6) = in(6) \setminus \{ (y,l) \mid l \in Lab \} \cup \{ (y,6) \}$

$F: (\wp(Var \times Lab))^{12} \rightarrow (\wp(Var \times Lab))^{12}$

System of Equations

In(1)	$\emptyset$				
In(2)	$\emptyset$				
In(3)	$\emptyset$				
In(4)	$\emptyset$				
In(5)	$\emptyset$				
In(6)	$\emptyset$				
Out(1)	$\emptyset$				
Out(2)	$\emptyset$				
Out(3)	$\emptyset$				
Out(4)	$\emptyset$				
Out(5)	$\emptyset$				
Out(6)	$\emptyset$				

$F: (\wp(Var \times Lab))^{12} \rightarrow (\wp(Var \times Lab))^{12}$

$in(1)=\{(x,?),(y,?),(z,?)\}$, $in(2)=out(1)$, $in(3)=out(2) \cup out(5)$, $in(4)=out(3)$, $in(5)=out(4)$, $In(6) = out(3)$

$out(1) = in(1) \setminus \{ (y,l) \mid l \in Lab \} \cup \{ (y,1) \}$, $out(2) = in(2) \setminus \{ (z,l) \mid l \in Lab \} \cup \{ (z,2) \}$

$out(3) = in(3)$, $out(4) = in(4) \setminus \{ (z,l) \mid l \in Lab \} \cup \{ (z,4) \}$

$out(5) = in(5) \setminus \{ (y,l) \mid l \in Lab \} \cup \{ (y,5) \}$, $out(6) = in(6) \setminus \{ (y,l) \mid l \in Lab \} \cup \{ (y,6) \}$

System of Equations

	$RD_\emptyset$	$F(RD_\emptyset)$	$F(F(RD_\emptyset))$	$F(F(F(RD_\emptyset)))$	$F(F(F(F(RD_\emptyset))))$
In(1)	∅	{(x,?),(y,?),(z,?)}	==	==	==
In(2)	∅	∅	{(y,1)}	{(x,?),(y,1),(z,?)}	==
In(3)	∅	∅	{(z,2),(y,5)}	{(z,2),(y,5)}	{(z,2),(z,4),(y,5)}
In(4)	∅	∅	∅	∅	{(z,2),(y,5)}
In(5)	∅	∅	{(z,4)}	{(z,4)}	{(z,4)}
In(6)	∅	∅	∅	∅	{(z,2),(y,5)}
Out(1)	∅	{(y,1)}	{(x,?),(y,1),(z,?)}	==	==
Out(2)	∅	{(z,2)}	{(z,2)}	{(z,2),(y,1)}	==
Out(3)	∅	∅	∅	{(z,2),(y,5)}	{(z,2),(y,5)}
Out(4)	∅	{(z,4)}	{(z,4)}	{(z,4)}	{(z,4)}
Out(5)	∅	{(y,5)}	{(y,5)}	{(z,4),(y,5)}	{(z,4),(y,5)}
Out(6)	∅	{(y,6)}	{(y,6)}	{(y,6)}	{(y,6)}

...

$F: (\wp(\text{Var} \times \text{Lab}))^{12} \rightarrow (\wp(\text{Var} \times \text{Lab}))^{12}$

$RD \sqsubseteq RD'$ when $\forall i$: $RD(i) \subseteq RD'(i)$

Monotonicity

F: $(\wp(\text{Var} \times \text{Lab}))^{12} \rightarrow (\wp(\text{Var} \times \text{Lab}))^{12}$

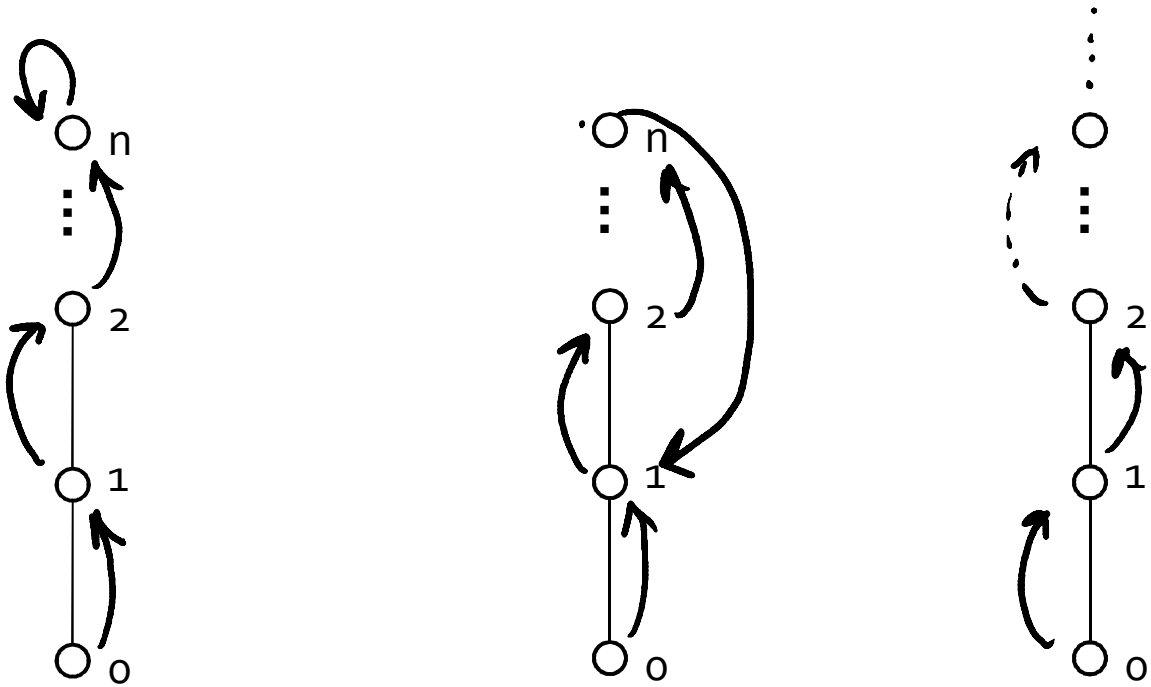
	$RD_{\emptyset}$	$F(RD_{\emptyset})$	$F(F(RD_{\emptyset}))$	$F(F(F(RD_{\emptyset})))$	$F(F(F(F(RD_{\emptyset}))))$
…					
Out(1)	$\emptyset$	**{(y,1)}**	**{(x,?),(y,1),(z,?)}**	**{(y,1)}**	**….**
…					

…

out(1) = {(y,1)} when (x,?) ∈ out(1)
in(1) \ { (y,l) | l ∈ Lab } U { (y,1) }, otherwise

(silly example just for illustration)

Convergence



Least Fixed Point

- We will see later why it exists
- For now, mostly informally…

$F: (\wp(\text{Var} \times \text{Lab}))^{12} \rightarrow (\wp(\text{Var} \times \text{Lab}))^{12}$

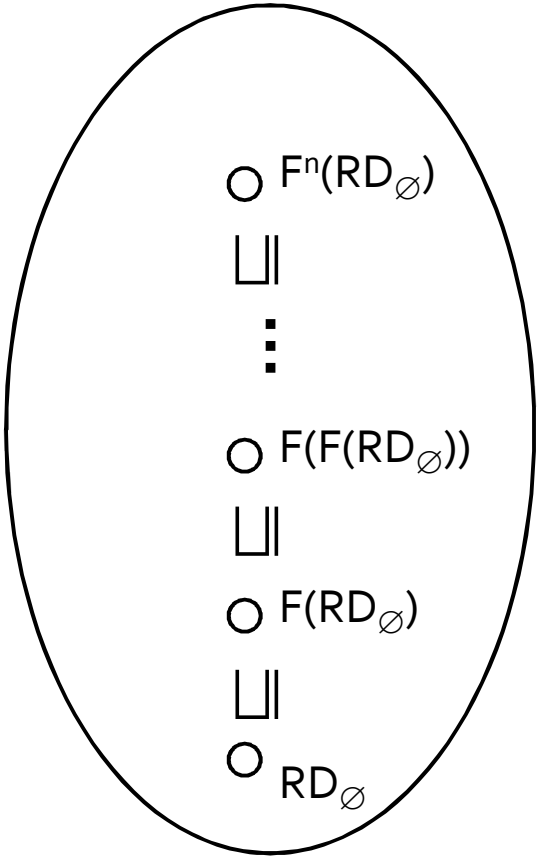
$RD \sqsubseteq RD'$ when $\forall i: RD(i) \subseteq RD'(i)$

F is monotone:
$RD \sqsubseteq RD'$ implies that $F(RD) \sqsubseteq F(RD')$

$RD_{\emptyset} = (\emptyset, \emptyset, \ldots, \emptyset)$

$F(RD_{\emptyset}), F(F(RD_{\emptyset}), F(F(F(RD_{\emptyset})), \ldots F^{n}(RD_{\emptyset})$

$F^{n+1}(RD_{\emptyset}) = F^{n}(RD_{\emptyset})$



Partially Ordered Sets (posets)

- Partial ordering is a relation
  $\sqsubseteq : L \times L \rightarrow \{ \text{true, false} \}$ that is:
  - Reflexive ($\forall l: l \sqsubseteq l$)
  - Transitive ($\forall l_1, l_2, l_3: l_1 \sqsubseteq l_2 \wedge l_2 \sqsubseteq l_3 \Rightarrow l_1 \sqsubseteq l_3$)
  - Anti-symmetric ($l_1 \sqsubseteq l_2 \wedge l_2 \sqsubseteq l_1 \Rightarrow l_1 = l_2$)

- A partially ordered set $(L, \sqsubseteq)$ is a set L equipped with a partial ordering $\sqsubseteq$

- For example: $(\wp(S), \subseteq)$

- Captures intuition of implication between facts
  - $l_1 \sqsubseteq l_2$ will intuitively mean $l_1 \Rightarrow l_2$
  - Later, in abstract domains , when $l_1 \sqsubseteq l_2$, we will say that l_1 is "more precise" than l_2 (represents fewer concrete states)

Bounds in Posets

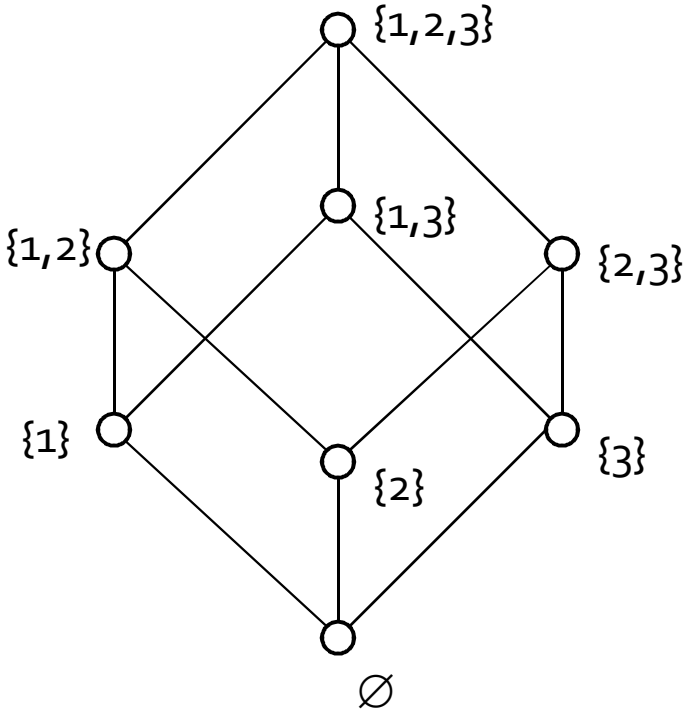
- Given a poset $(L, \sqsubseteq)$
- A subset Y of L has $l \in L$ as an upper bound if $\forall l' \in Y : l' \sqsubseteq l$
- A subset Y of L has $l \in L$ as a lower bound if $\forall l' \in Y : l' \sqsupseteq l$ (we write $l' \sqsupseteq l$ when $l \sqsubseteq l'$)
- A least upper bound (lub) l of Y is an upper bound of Y that satisfies $l \sqsubseteq l'$ whenever l' is another upper bound of Y
- A greatest lower bound (glb) l of Y is a lower bound of Y that satisfies $l \sqsupseteq l'$ whenever l' is another lower bound of Y

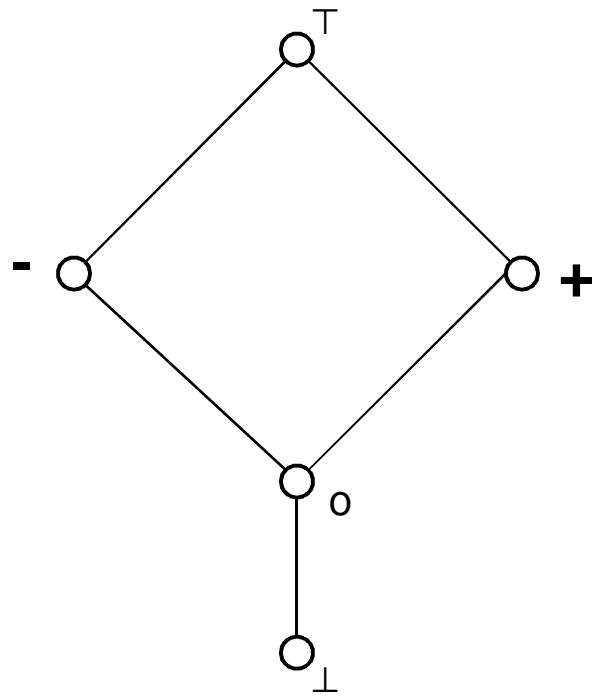
- For any subset $Y \subseteq L$
  - If the lub exists then it is unique, and denoted by $\sqcup Y$ (join)
    - We often write $l_1 \sqcup l_2$ for $\sqcup\{ l_1, l_2\}$
  - If the glb exists then it is unique, and denoted by $\sqcap Y$ (meet)
    - We often write $l_1 \sqcap l_2$ for $\sqcap\{ l_1, l_2\}$

Complete Lattices

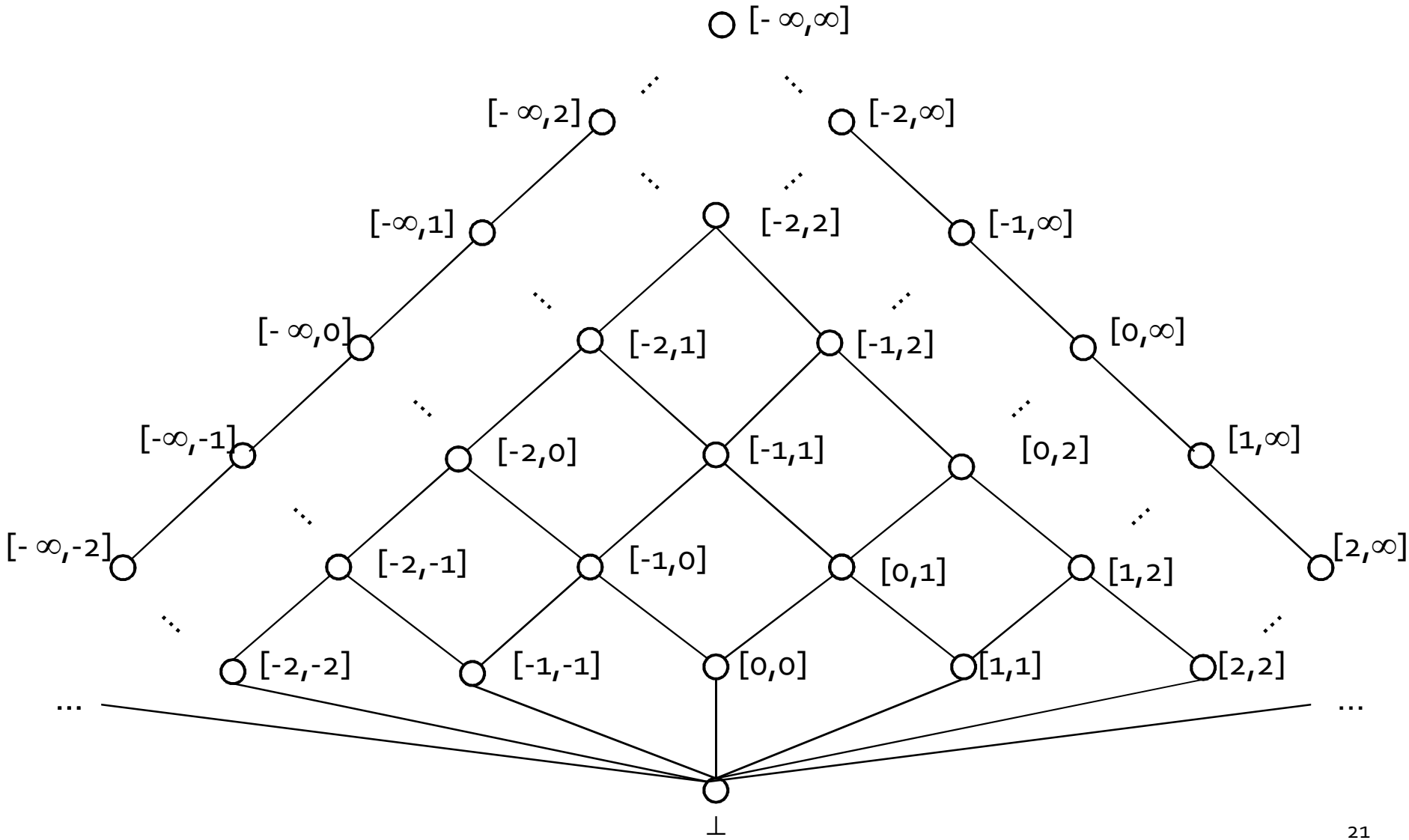
- A complete lattice $(L, \sqsubseteq)$ is a poset such that all subsets have least upper bounds as well as greatest lower bounds
- In particular
  - $\bot = \bigsqcup\emptyset = \bigsqcap L$ is the least element (bottom)
  - $\top = \bigsqcup L = \bigsqcap\emptyset$ is the greatest element (top)

Complete Lattices: Examples





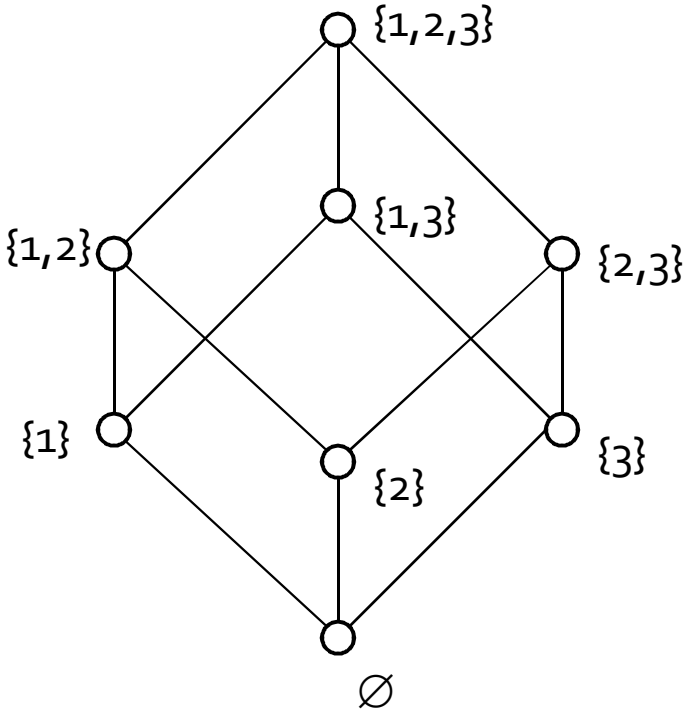
Complete Lattices: Examples

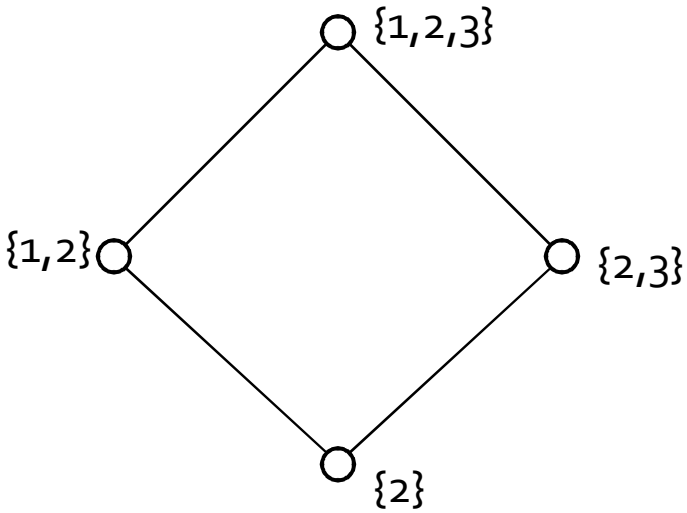


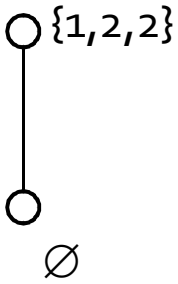
Moore Families

- A Moore family is a subset Y of a complete lattice $(L,\sqsubseteq)$ that is closed under greatest lower bounds
  - $\forall Y' \subseteq Y: \sqcap Y' \in Y$
  - Hence, a Moore family is never empty
  - Always contains least element $\sqcap Y$ and a greatest element $\sqcap \emptyset$
- Why do we care?
  - We will see that Moore families correspond to choices of abstractions (as targets of upper closure operators)
  - More on this… later…

Moore Families: Examples







Chains

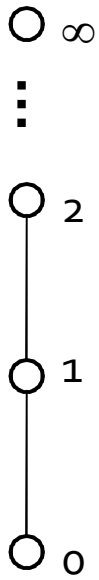
- In program analysis we will often look at sequences of properties of the form $l_0 \sqsubseteq l_1 \sqsubseteq \ldots \sqsubseteq l_n$

- Given a poset $(L, \sqsubseteq)$ a subset $Y \subseteq L$ is a chain if every two elements in Y are ordered
  - $\forall l_1, l_2 \in Y$: $(l_1 \sqsubseteq l_2)$ or $(l_2 \sqsubseteq l_1)$
  - When Y is a finite subset we say that the chain is a finite chain
- A sequence of elements $l_0 \sqsubseteq l_1 \sqsubseteq \ldots \in L$ is an ascending chain
- A sequence of elements $l_0 \sqsupseteq l_1 \sqsupseteq \ldots \in L$ is a descending chain
- L has finite height when all chains in L are finite
- A poset $(L, \sqsubseteq)$ satisfies the ascending chain condition (ACC) if every ascending chain eventually stabilizes, that is $\exists s \in N: \forall n \in N: n \geqslant s \Rightarrow l_n = l_s$

Chains: Example Posets

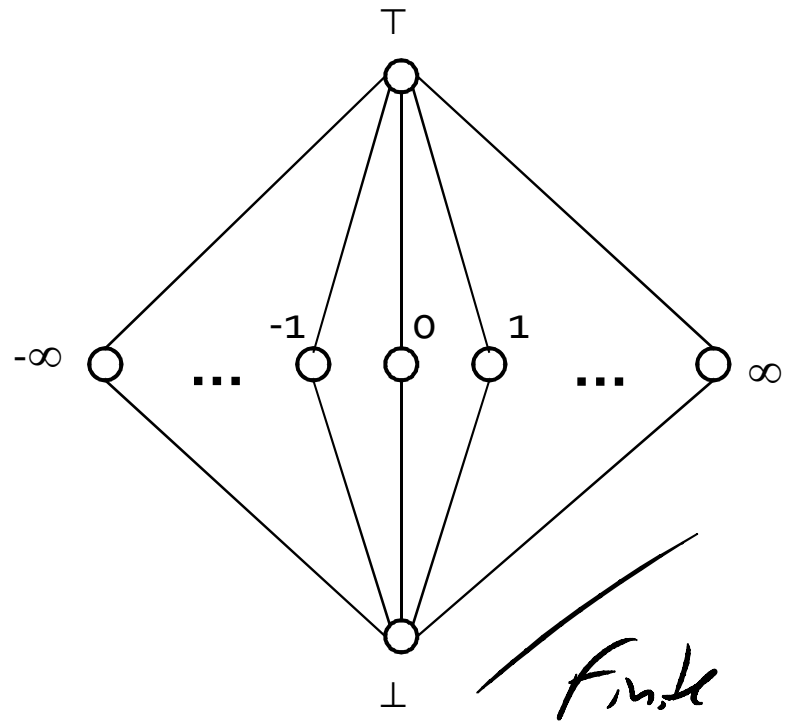
ACC

(a)



¬ACC

(b)



(c)

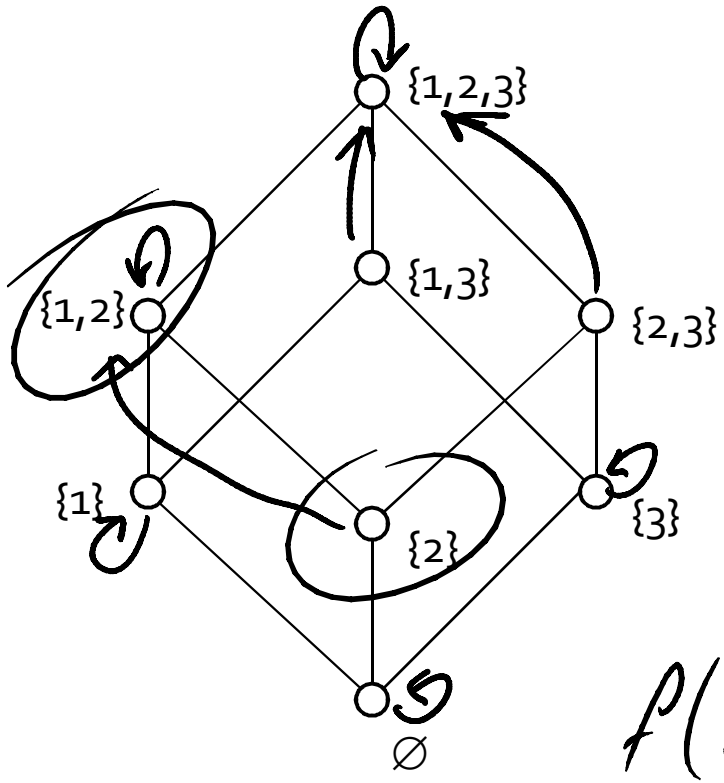
Finite Height

Properties of Functions

- A function $f: L1 \longrightarrow L2$ between two posets $(L1, \sqsubseteq 1)$ and $(L2, \sqsubseteq 2)$ is
- Monotone (order-preserving)
  - $\forall l, l' \in L1: l \sqsubseteq 1\ l' \Rightarrow f(l) \sqsubseteq 2\ f(l')$
  - Special case $f: L \longrightarrow L$, $\forall l, l' \in L: l \sqsubseteq l' \Rightarrow f(l) \sqsubseteq f(l')$

- Distributive (join morphism)
  - $\forall l, l' \in L1\ f(l \sqcup l') = f(l) \sqcup f(l')$

Function Properties: Examples



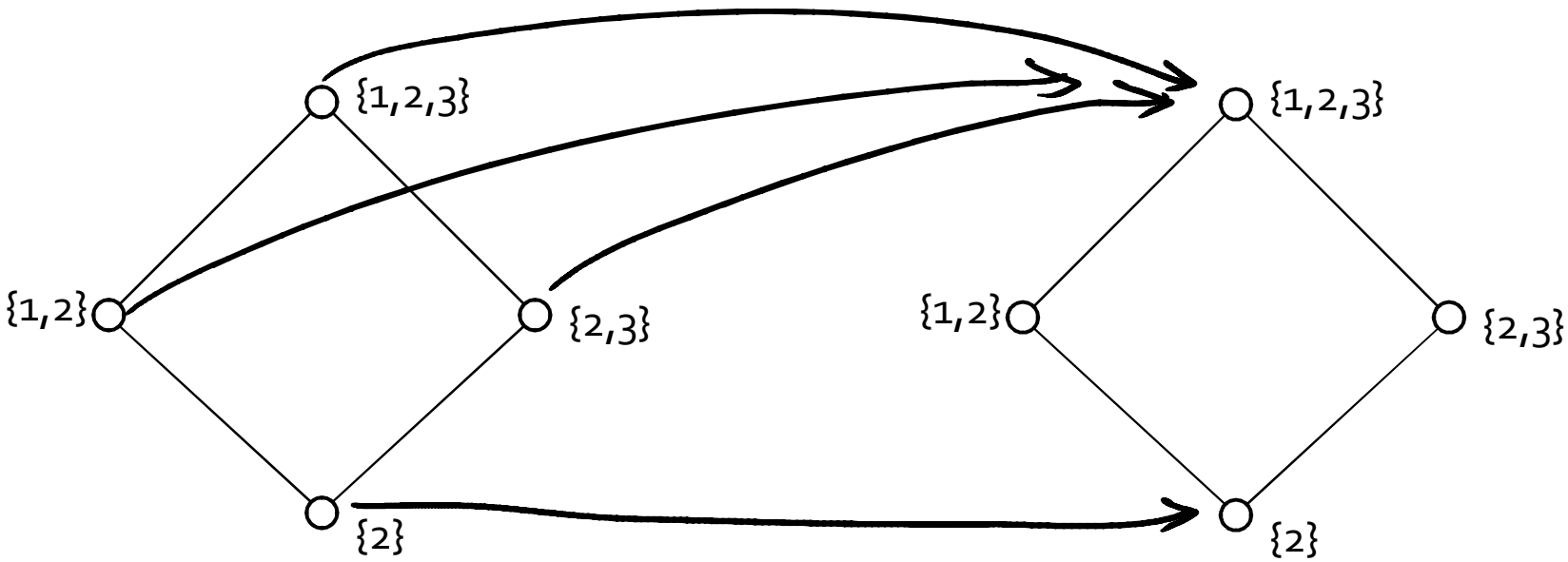
$\{2\} \subseteq \{1,2\}$

$\{1,2\} \subseteq \{1,2\}$

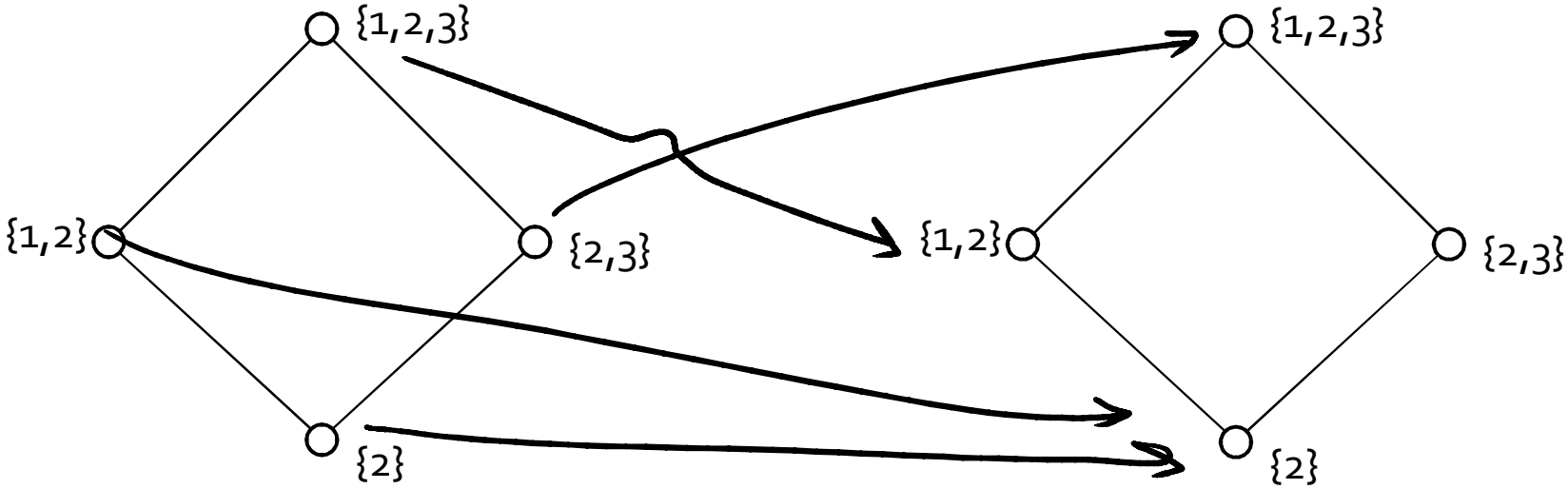
$f(\{2\}) = \{1,2\}$

$f(\{1,2\}) = \{1,2\}$

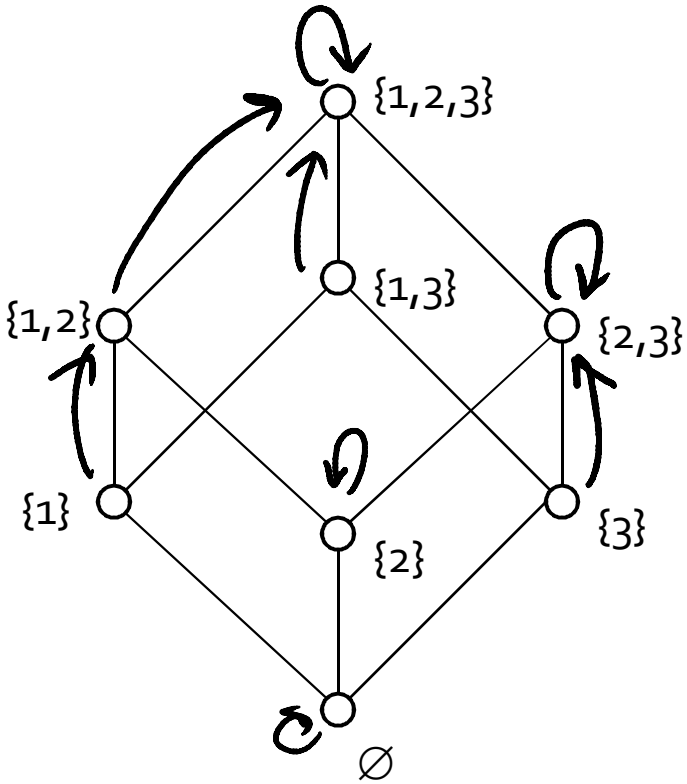
Function Properties: Examples

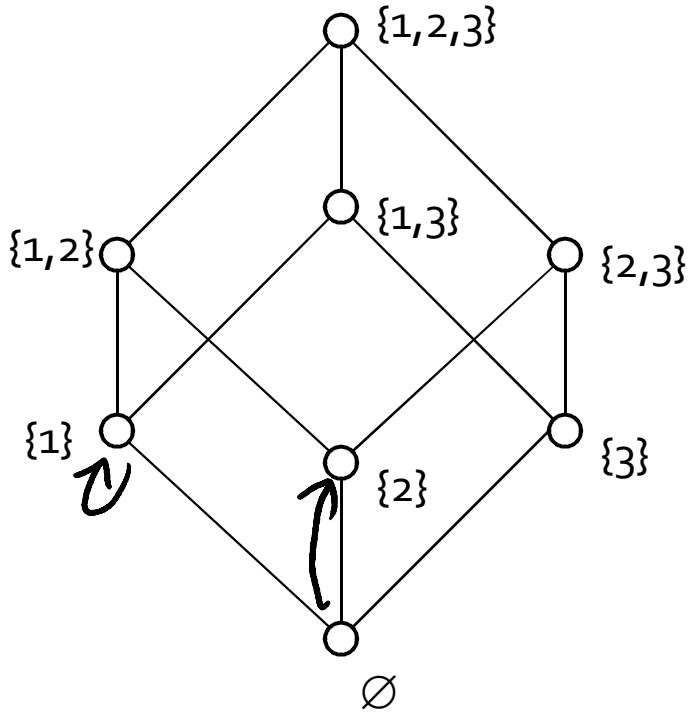


Function Properties: Examples



Function Properties: Examples





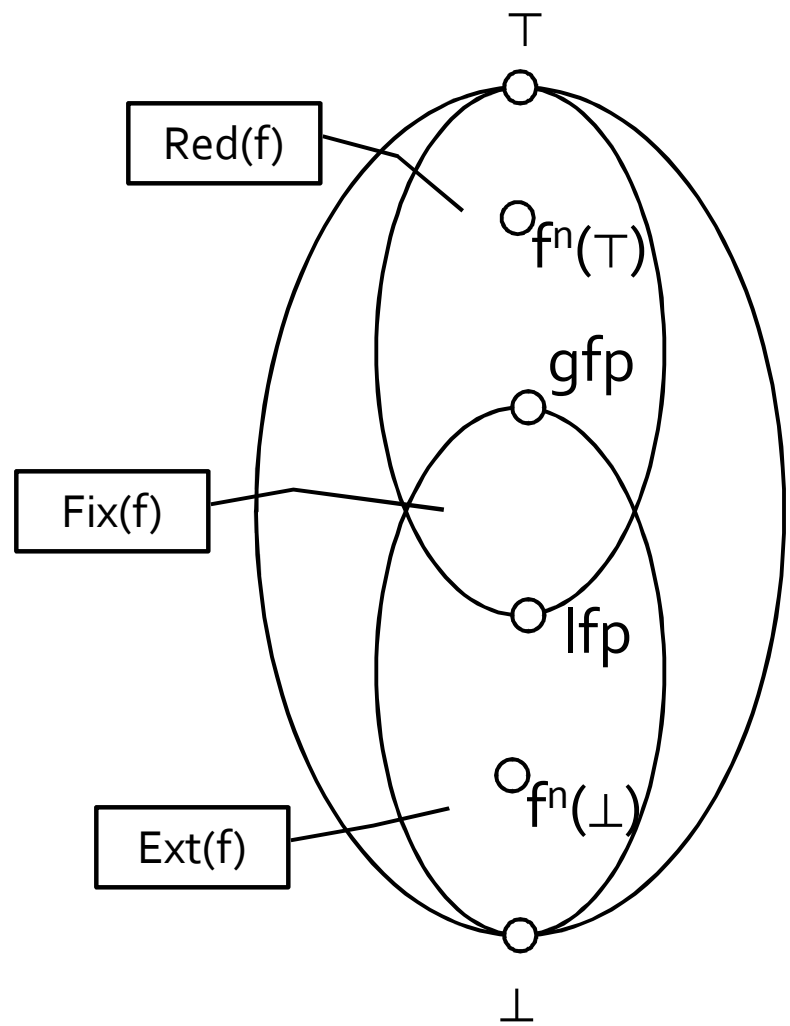
$f(\{1\}) \cup f(\emptyset) = \{1,2\}$

$f(\{1\} \cup \emptyset) = f(\{1\}) = \{1\}$

Fixed Points

- Consider a complete lattice $L = (L, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$ and a monotone function $f: L \rightarrow L$
- An element $l \in L$ is
  - A fixed point iff $f(l) = l$
  - A pre-fixedpoint iff $l \sqsubseteq f(l)$
  - A post-fixedpoint iff $l \sqsupseteq f(l)$

- Fix(f) = set of all fixed points
- Ext(f) = set of all pre-FP
- Red(f) = set of all post-FP



Tarski's Theorem

- Consider a complete lattice $L = (L, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$ and a monotone function $f: L \longrightarrow L$, then lfp(f) and gfp(f) exist and
  - $\text{lfp}(f) = \sqcap \text{Red}(f) = \sqcap \text{Fix}(f)$
  - $\text{gfp}(f) = \sqcup \text{Ext}(f) = \sqcup \text{Fix}(f)$

Kleene's Fixedpoint Theorem

- Consider a complete lattice $L = (L, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$ and a continuous function $f: L \rightarrow L$ then

  $lfp(f) = \bigsqcup_{n \in N} f^n(\bot)$

- A function f is continuous if for every increasing chain $Y \subseteq L$, $f(\sqcup Y) = \sqcup \{ f(y) \mid y \in Y \}$

- Monotone functions on posets satisfying ACC are continuous
  - Every ascending chain eventually stabilizes
    $l_0 \sqsubseteq l_1 \sqsubseteq \ldots \sqsubseteq l_n = l_{n+1} = \ldots$
    hence l_n is the least upper bound of $\{ l_0, l_1, \ldots, l_n \}$, thus $f(\sqcup Y) = f(l_n)$
  - From monotonicity of f
    $f(l_0) \sqsubseteq f(l_1) \sqsubseteq \ldots \sqsubseteq f(l_n) = f(l_{n+1}) = \ldots$
    Hence $f(l_n)$ is the least upper bound of $\{ f(l_0), f(l_1), \ldots, f(l_n) \}$, thus
    $\sqcup \{ f(y) \mid y \in Y \} = f(l_n)$

An Algorithm for Computing lfp

- Kleene's fixed point theorem gives a constructive method for computing the lfp

$$\mathrm{lfp}(f) = \bigsqcup_{n \in N} f^n(\bot)$$

```
l = ⊥
while f(l) ≠ l do l = f(l)
```

- In our case, can compute efficiently using a worklist algorithm for "*chaotic iteration*"

Finally…

Reaching Definitions…

in(1) = { (x,?), (y,?), (z,?) }
in(2) = out(1)
in(3) = out(2) U out(5)
in(4) = out(3)
in(5) = out(4)
In(6) = out(3)
out(1) = in(1) \ { (y,l) | l ∈ Lab } U { (y,1) }
out(2) = in(2) \ { (z,l) | l ∈ Lab } U { (z,2) }
out(3) = in(3)
out(4) = in(4) \ { (z,l) | l ∈ Lab } U { (z,4) }
out(5) = in(5) \ { (y,l) | l ∈ Lab } U { (y,5) }
out(6) = in(6) \ { (y,l) | l ∈ Lab } U { (y,6) }

$$F: (\wp(\text{Var} \times \text{Lab}))^{12} \rightarrow (\wp(\text{Var} \times \text{Lab}))^{12}$$

$$\boxed{\begin{array}{l} l = \bot \\ \text{while } F(l) \neq l \text{ do } l = F(l) \end{array}}$$

Algorithm Revisited

- Input: equation system for RD
- Output: lfp(F)
- Algorithm:
  - $RD_1 = \emptyset, \ldots, RD_{12} = \emptyset$
  - While $RD_j \neq F_j(RD_1,\ldots,RD_{12})$ for some j
    - $RD_j = F_j(RD_1,\ldots,RD_{12})$

- Idea: non-deterministically select which component of RD should make a step
- Can make selection efficiently based on dependencies
  - We have information on what equations depend on Fj that has been updated, need only recompute these on every step

But…

- where did the transfer functions come from?
  - e.g., $out(1) = in(1) \setminus \{ (y,l) \mid l \in Lab \} \cup \{ (y,1) \}$
- how do we know transfer functions really over-approximate the concrete operations?

Trace Semantics

$[y := x]^1;$
$[z := 1]^2;$
$\text{while } [y > 0]^3 \ ($
$\quad [z := z * y]^4;$
$\quad [y := y - 1]^5;$
$)$
$[y := 0]^6$

$$< 1,\{ x\mapsto 42, y\mapsto 0, z\mapsto 0 \} > \overset{[y := x]^1}{\Rightarrow} < 2,\{ x\mapsto 42, y\mapsto 42, z\mapsto 0 \} > \overset{[z := 1]^2}{\Rightarrow}$$

$$< 3,\{ x\mapsto 42, y\mapsto 42, z\mapsto 1 \} > \overset{[y > 0]^3}{\Rightarrow} < 4,\{ x\mapsto 42, y\mapsto 42, z\mapsto 1 \}> \ldots$$

$$< 1,\{ x\mapsto 73, y\mapsto 0, z\mapsto 0 \} > \overset{[y := x]^1}{\Rightarrow} < 2,\{ x\mapsto 73, y\mapsto 73, z\mapsto 0 \} > \overset{[z := 1]^2}{\Rightarrow}$$

$$< 3,\{ x\mapsto 73, y\mapsto 73, z\mapsto 1 \} > \overset{[y > 0]^3}{\Rightarrow} < 4,\{ x\mapsto 73, y\mapsto 73, z\mapsto 1 \}> \ldots$$

$$\vdots$$

- note that input (x) is unknown
- while loop?
- trace semantics is not computable

Instrumented Trace Semantics

- on every assignment, record the label in which the assignment was performed

Abstract Interpretation

```
1: y := x;
2: z := 1;
3: while y > 0 {
4:    z := z * y;
5:    y := y – 1
 }
6: y := 0
```

(x,?)(y,?)(z,?)(y,1)(z,2)(y,6)

(x,?)(y,?)(z,?)(y,1)(z,2)(z,4)(y,5)(y,6)

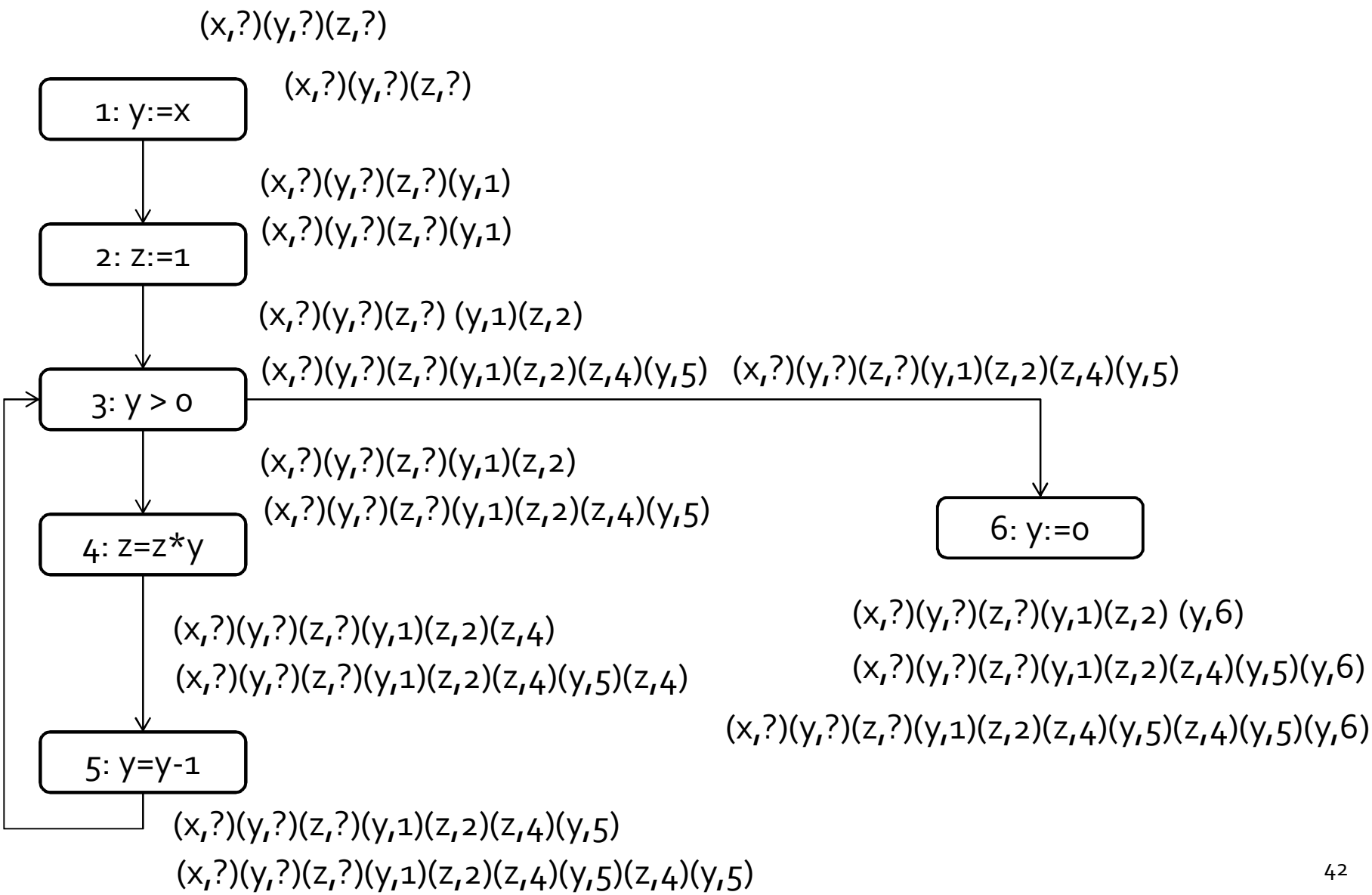
(x,?)(y,?)(z,?)(y,1)(z,2)(z,4)(y,5)(z,4)(y,5)(y,6)

(x,?)(y,?)(z,?)(y,1)(z,2)(z,4)(y,5)(z,4)(y,5)(z,4)(y,5)(y,6)

⋮

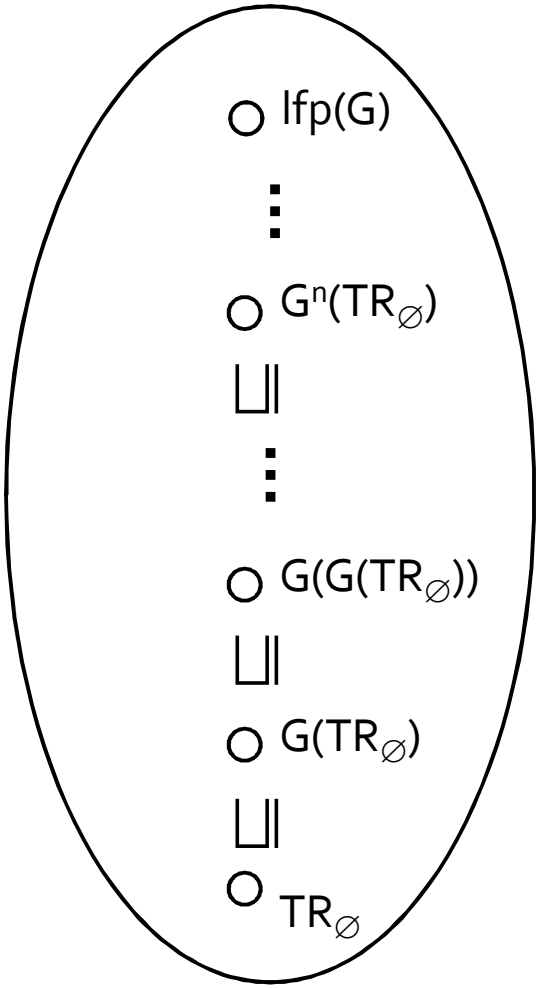
- Instrumented trace semantics

Collecting Semantics



System of Equations

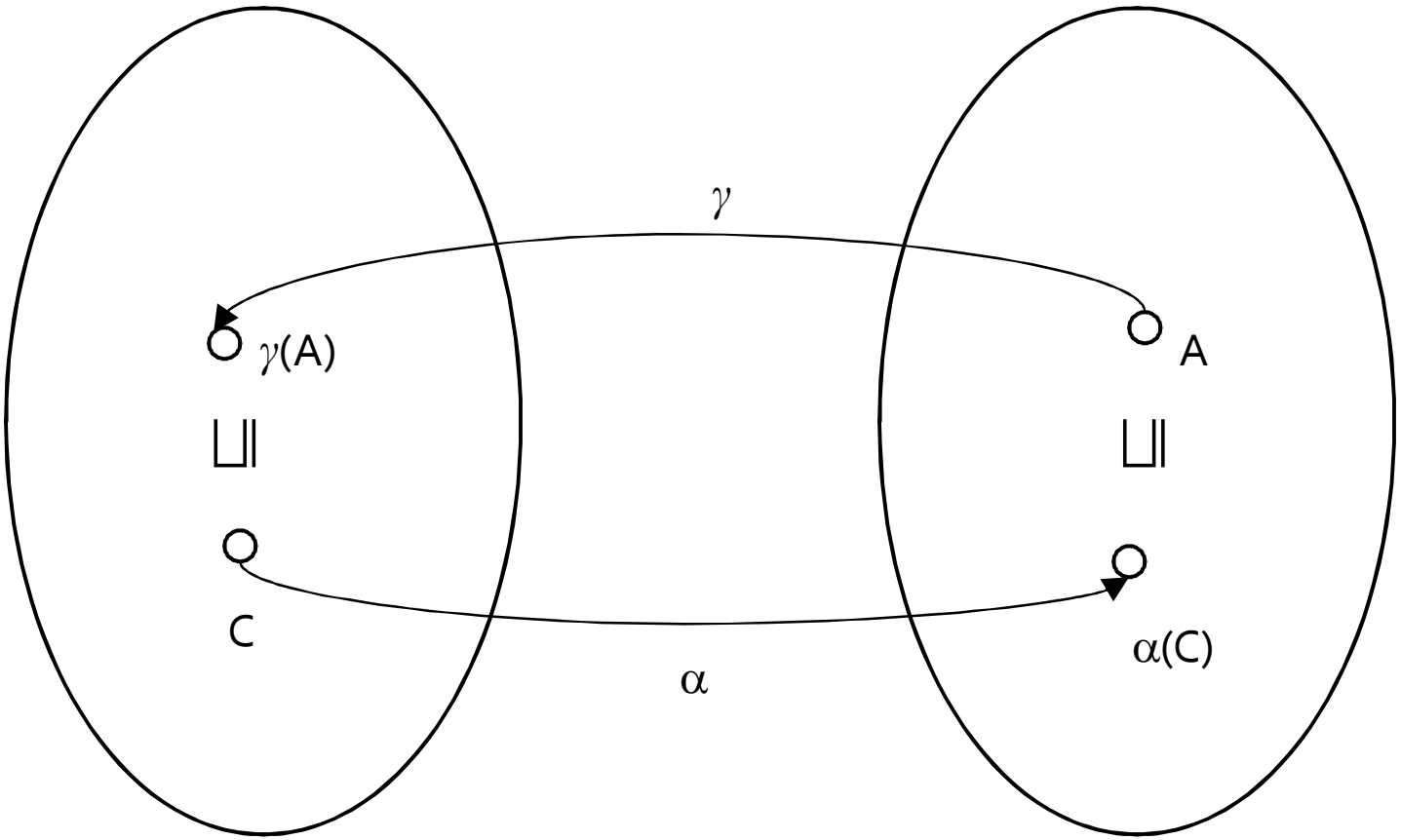
in(1) = { (x,?)(y,?)(z,?) }
in(2) = out(1)
in(3) = out(2) U out(5)
in(4) = out(3)
in(5) = out(4)
In(6) = out(3)
out(1) = { $\pi \bullet$ (y,1) | $\pi \in$ in(1) }
out(2) = { $\pi \bullet$ (z,2) | $\pi \in$ in(2) }
out(3) = in(3)
out(4) = { $\pi \bullet$ (z,4) | $\pi \in$ in(4) }
out(5) = { $\pi \bullet$ (y,5) | $\pi \in$ in(5) }
out(6) = { $\pi \bullet$ (y,6) | $\pi \in$ in(6) }



$\pi \in$ Trace = (Var x Lab)*

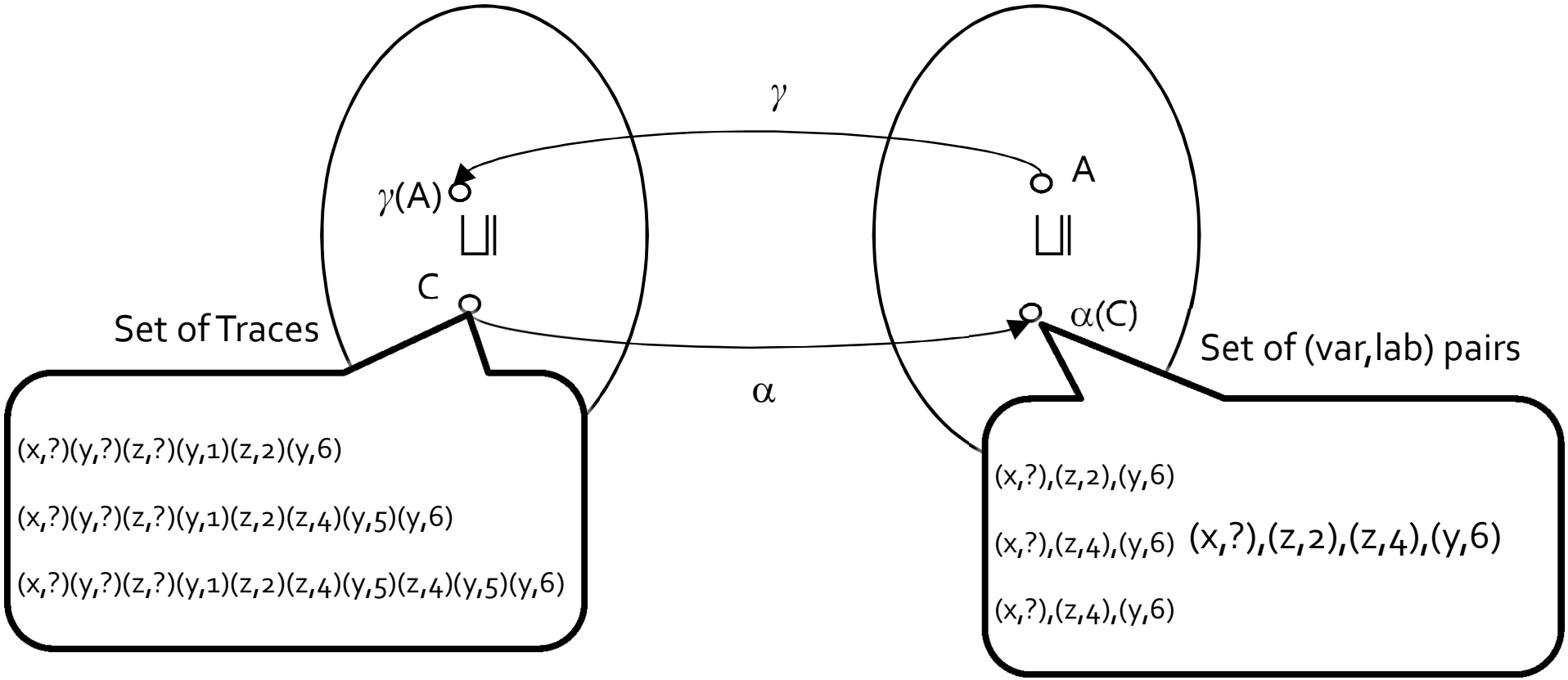
G: ($\wp$(Trace))^{12} $\rightarrow$ ($\wp$(Trace))^{12}

Galois Connection



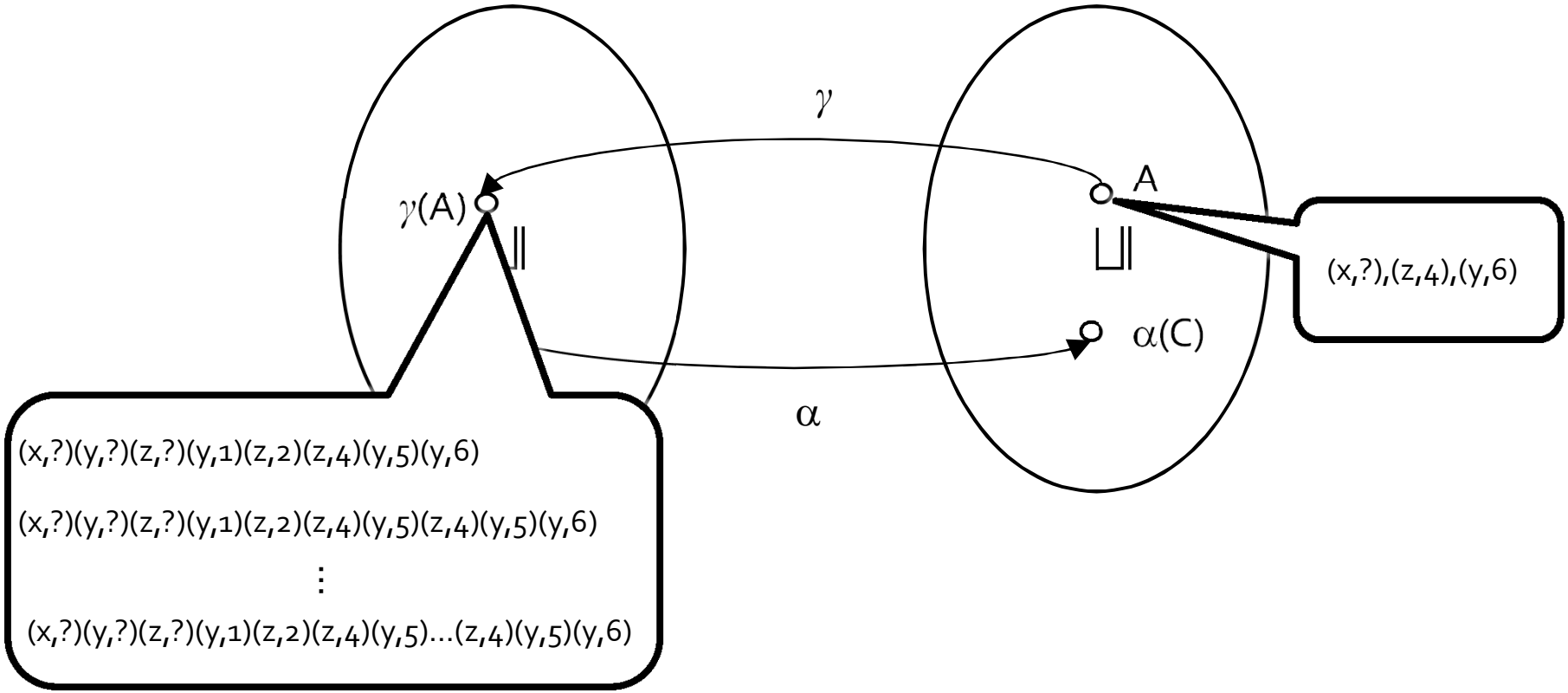
$$\alpha(C) \sqsubseteq A \Leftrightarrow C \sqsubseteq \gamma(A)$$

Reaching Definitions: Abstraction



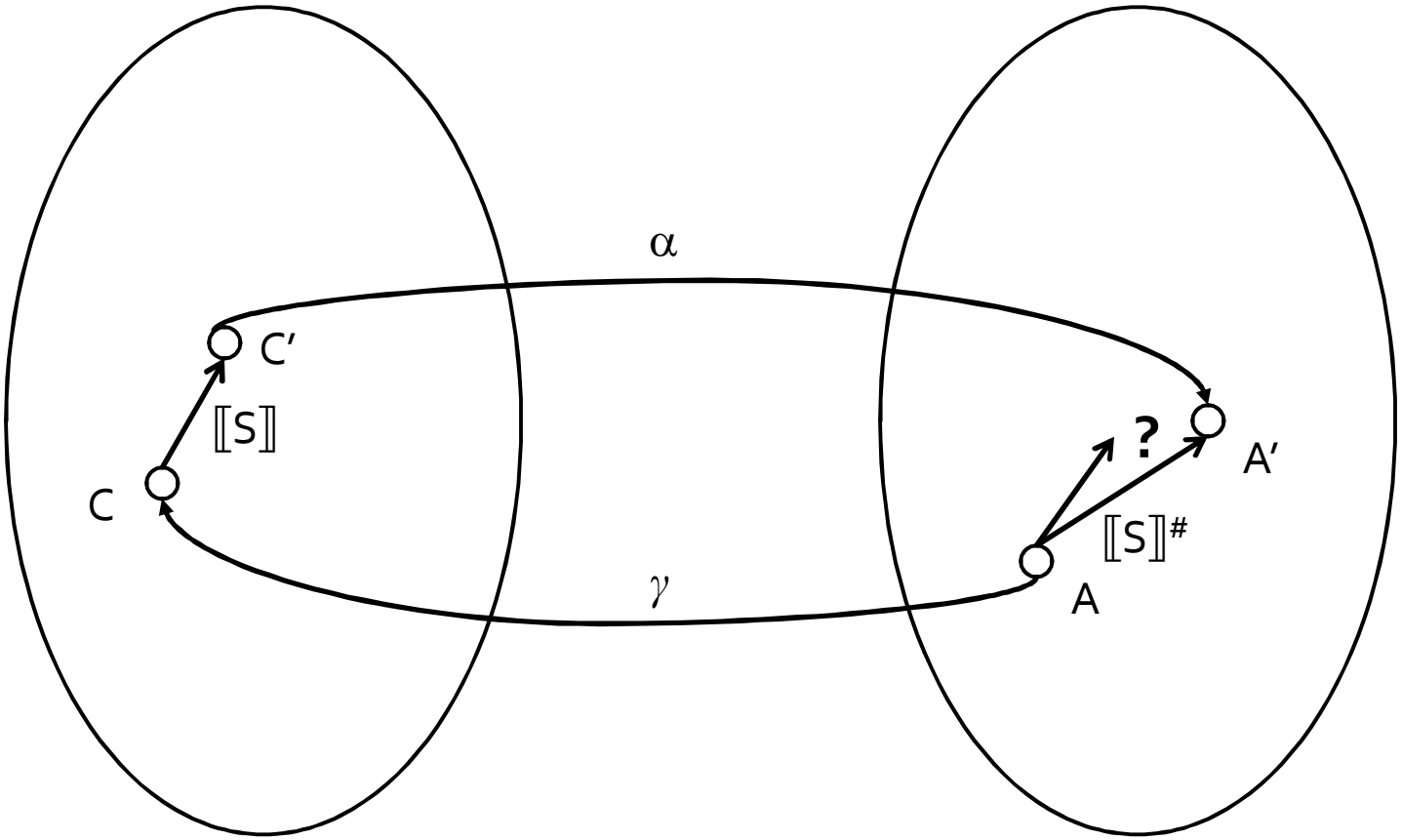
- DOM(π) = variables in π
- SRD(π)(v) = l iff rightmost pair for v in π is (v,l)
- $\alpha(C) = \{ (v, SRD(\pi)(v)) \mid v \in DOM(\pi) \wedge \pi \in C \}$
- $\gamma(A) = \{ \pi \mid \forall v \in DOM(\pi) : (v, SRD(\pi)(v)) \in A \}$

Reaching Definitions: Concretization



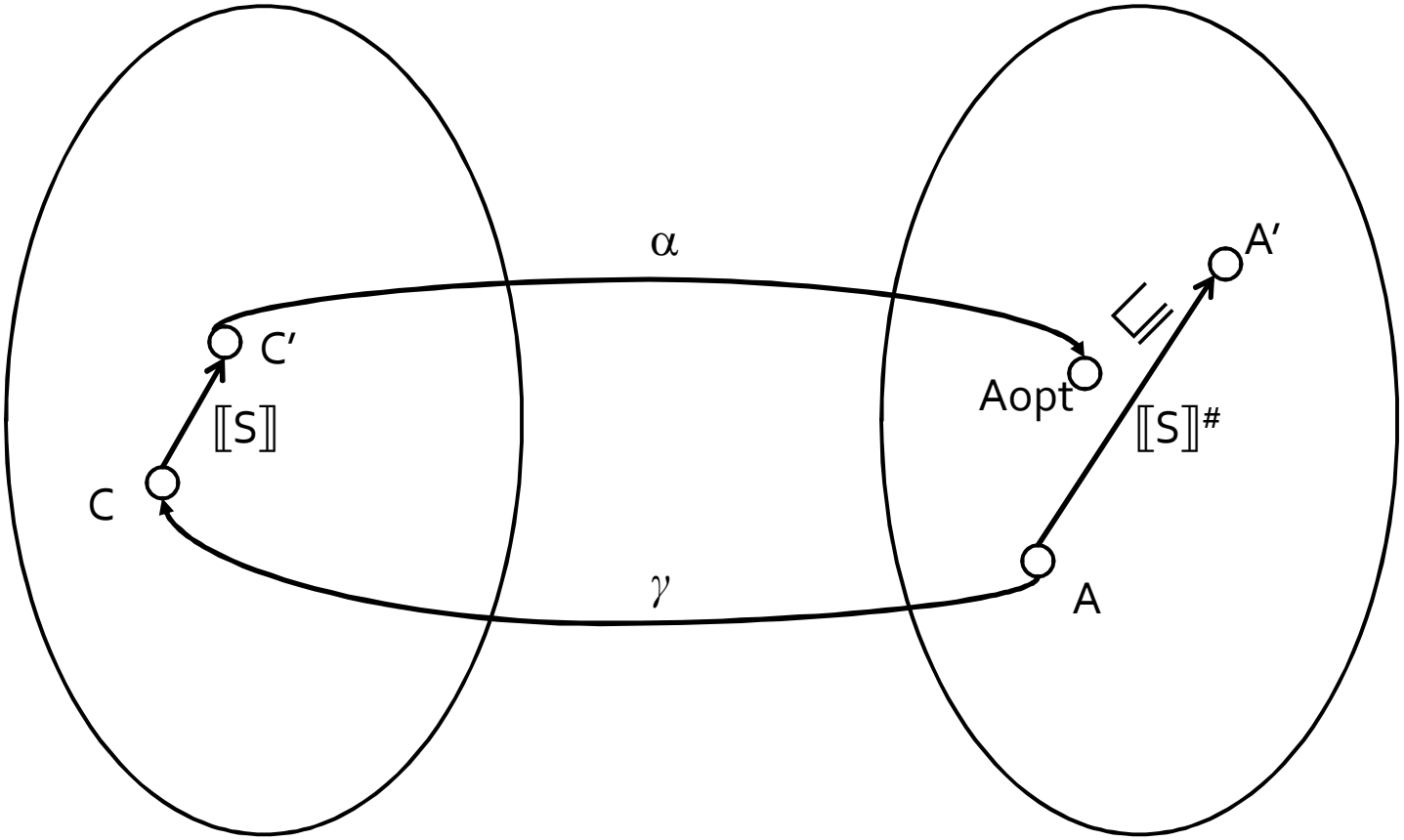
- DOM(π) = variables in π
- SRD(π)(v) = l iff rightmost pair for v in π is (v,l)
- $\alpha(C) = \{ (v, SRD(\pi)(v)) \mid v \in DOM(\pi) \wedge \pi \in C \}$
- $\gamma(A) = \{ \pi \mid \forall v \in DOM(\pi) : (v, SRD(\pi)(v)) \in A \}$

Best (Induced) Transformer



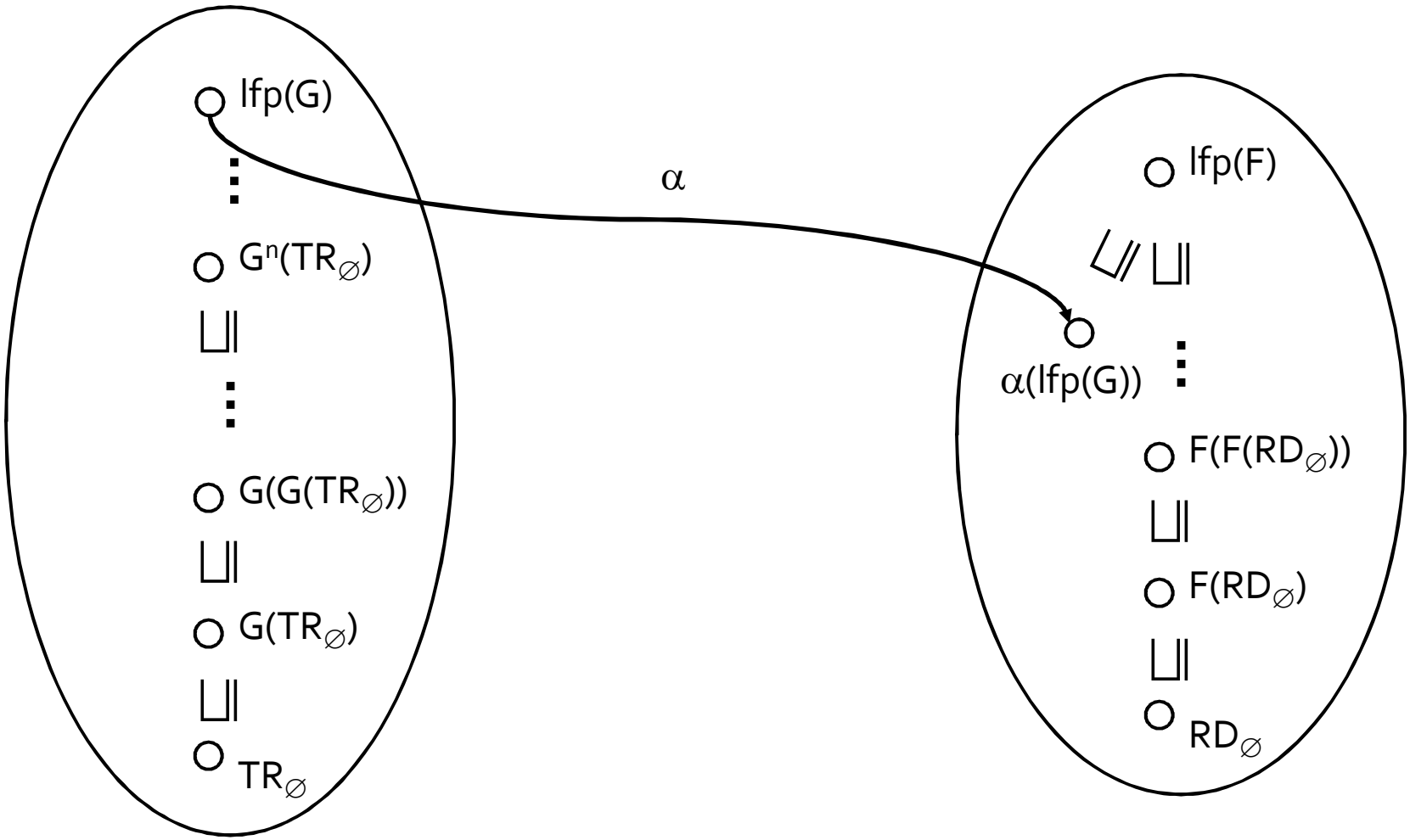
How do we figure out the abstract effect ⟦S⟧# of a statement S ?

Sound Abstract Transformer



$$\alpha \circ [\![S]\!] \circ \gamma(A) \sqsubseteq [\![S]\!]^{\#}(A)$$

Soundness of Induced Analysis



$$\alpha(\mathsf{lfp}(G)) \sqsubseteq \mathsf{lfp}(\alpha \circ G \circ \gamma) \sqsubseteq \mathsf{lfp}(F)$$

Back to a bit of dataflow analysis…

Recap

- Represent properties of a program using a lattice $(L, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$
- A continuous function $f: L \longrightarrow L$
  - Monotone function when L satisfies ACC implies continuous
- Kleene's fixedpoint theorem
  - $\text{lfp}(f) = \bigsqcup_{n \in N} f^n(\bot)$
- A constructive method for computing the lfp

Some required notation

blocks : Stmt $\longrightarrow$ P(Blocks)
blocks([x := a]^{lab}) = {[x := a]^{lab}}
blocks([skip]^{lab}) = {[skip]^{lab}}
blocks(S1; S2) = blocks(S1) $\cup$ blocks(S2)
blocks(if [b]^{lab} then S1 else S2) = {[b]^{lab}} $\cup$ blocks(S1) $\cup$ blocks(S2)
blocks(while [b]^{lab} do S) = {[b]^{lab}} $\cup$ blocks(S)

FV: (BExp $\cup$ AExp)$\longrightarrow$ Var
Variables used in an expression

AExp(a) = all non-unit expressions in the arithmetic expression a
similarly AExp(b) for a boolean expression b

Available Expressions Analysis

$[x := a+b]^1;$
$[y := a*b]^2;$
while $[y > a+b]^3$ (
  $[a := a + 1]^4;$
  $[x := a + b]^5$
)

(a+b) always available at label 3

For each program point, which expressions <u>must</u> have already been computed, and not later modified, on <u>all paths</u> to the program point

Available Expressions Analysis

- Property space
  - in_{AE}, out_{AE}: Lab $\rightarrow \wp$(AExp)
  - Mapping a label to set of arithmetic expressions available at that label

- Dataflow equations
  - Flow equations – how to join incoming dataflow facts
  - Effect equations - given an input set of expressions S, what is the effect of a statement

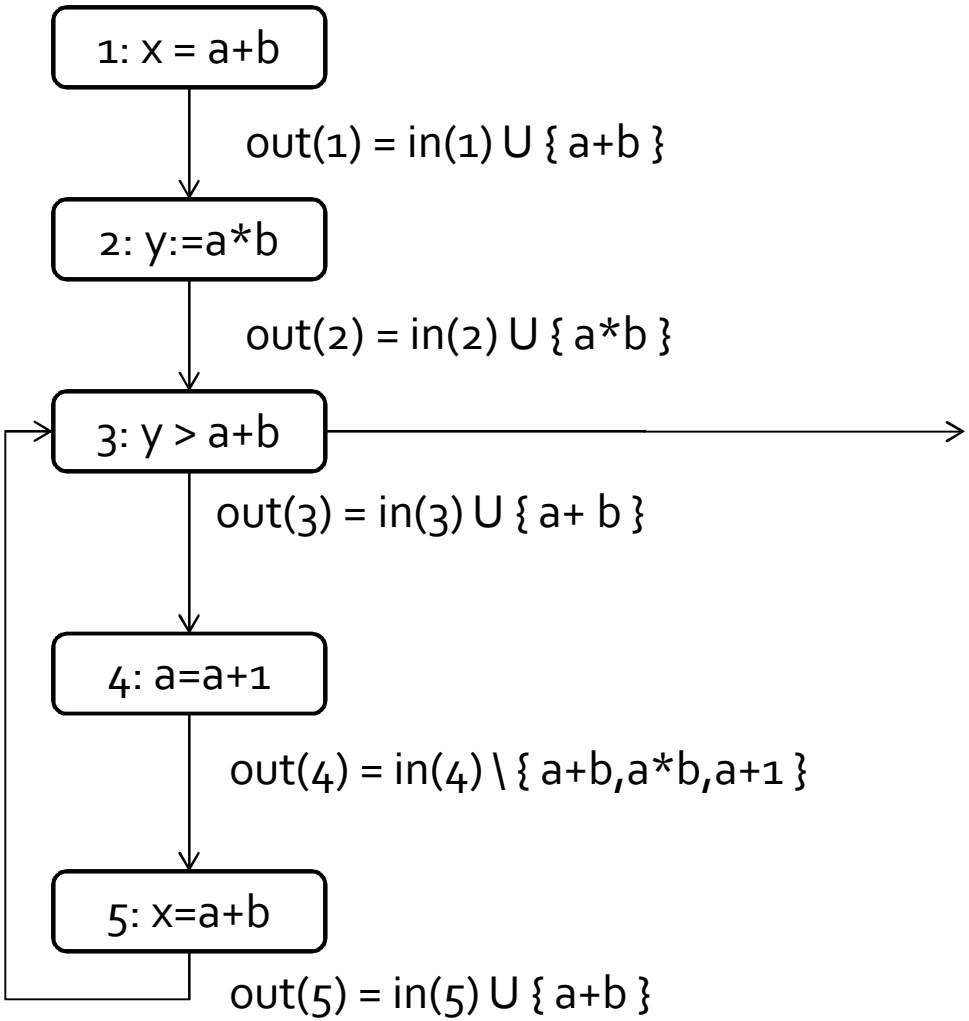
Available Expressions Analysis

- $in_{AE}(lab) =$
  - $\emptyset$ when lab is the initial label
  - $\bigcap \{ out_{AE}(lab') \mid lab' \in pred(lab) \}$ otherwise
- $out_{AE}(lab) = \ldots$

Block	**out (lab)**
$[x := a]^{lab}$	$in(lab) \setminus \{ a' \in AExp \mid x \in FV(a') \} \cup \{ a' \in AExp(a) \mid x \notin FV(a') \}$
$[skip]^{lab}$	$in(lab)$
$[b]^{lab}$	$in(lab) \cup AExp(b)$

From now on going to drop the AE subscript when clear from context

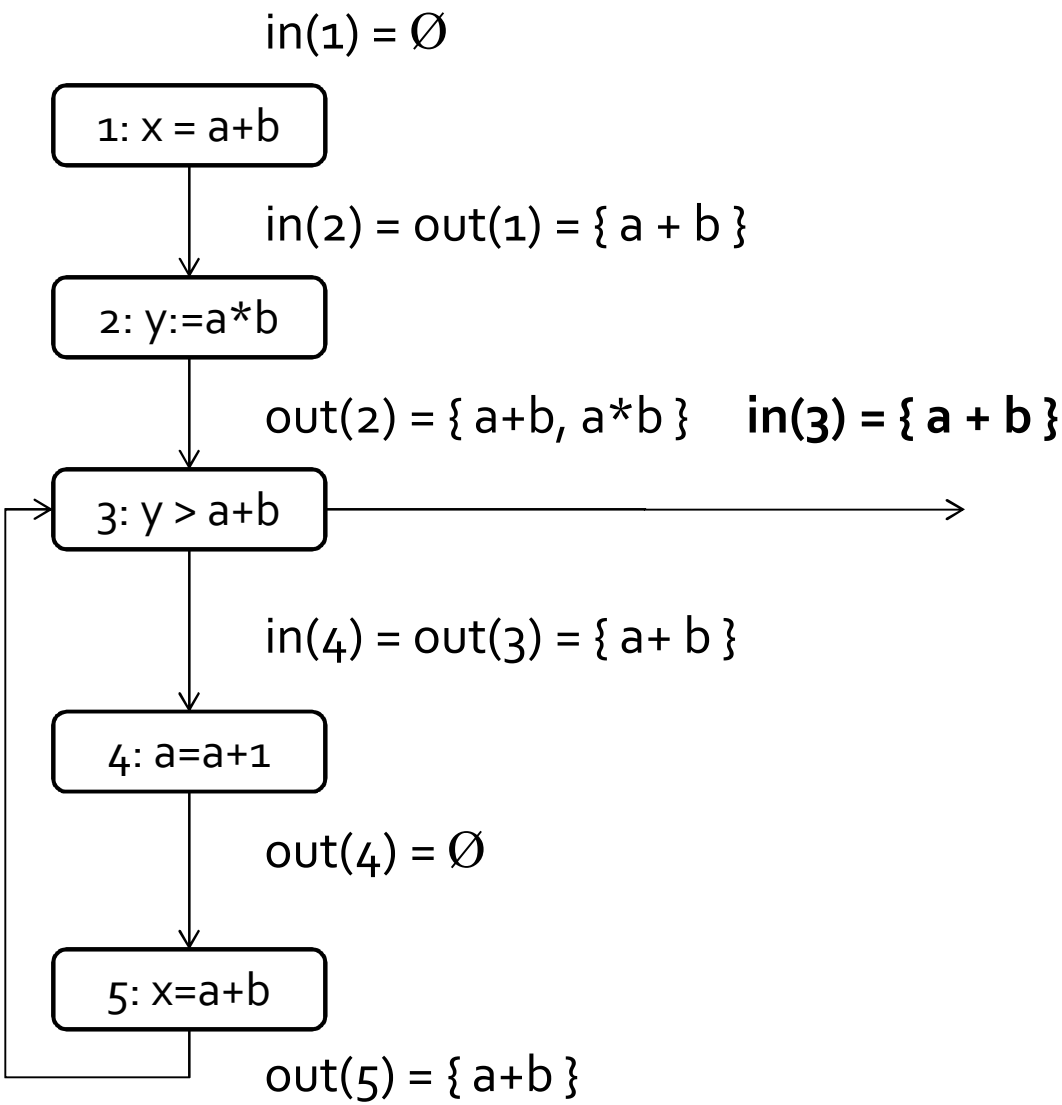
Transfer Functions



in(1) = Ø
in(2) = out(1)
in(3) = out(2) ∩ out(5)
in(4) = out(3)
in(5) = out(4)

[x := a+b]^{1};
[y := a*b]^{2};
while [y > a+b]^{3} (
  [a := a + 1]^{4};
  [x := a + b]^{5}
)

Solution


in(1) = Ø
1: x = a+b
in(2) = out(1) = { a + b }
2: y:=a*b
out(2) = { a+b, a*b }
in(3) = { a + b }
3: y > a+b
in(4) = out(3) = { a+ b }
4: a=a+1
out(4) = Ø
5: x=a+b
out(5) = { a+b }

Kill/Gen

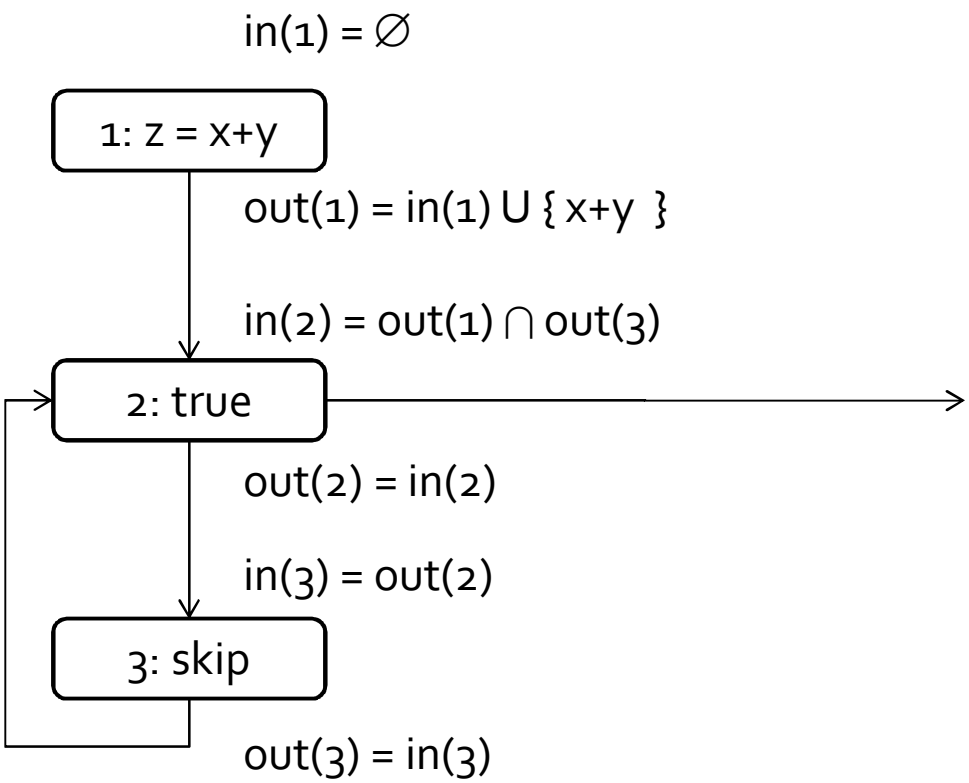
Block	**out (lab)**
$[x := a]^{lab}$	$in(lab) \setminus \{ a' \in AExp \mid x \in FV(a') \} \cup \{ a' \in AExp(a) \mid x \notin FV(a') \}$
$[skip]^{lab}$	$in(lab)$
$[b]^{lab}$	$in(lab) \cup AExp(b)$

Block	**kill**	**gen**
$[x := a]^{lab}$	$\{ a' \in AExp \mid x \in FV(a') \}$	$\{ a' \in AExp(a) \mid x \notin FV(a') \}$
$[skip]^{lab}$	$\emptyset$	$\emptyset$
$[b]^{lab}$	$\emptyset$	$AExp(b)$

$$out(lab) = in(lab) \setminus kill(B^{lab}) \cup gen(B^{lab})$$

B^{lab} = block at label lab

Why solution with largest sets?



in(1) = Ø
in(2) = out(1) ∩ out(3)
in(3) = out(2)

[z := x+y]^1;
while [true]^2 (
  [skip]^3;
)

After simplification: in(2) = in(2) ∩ { x+y }

Solutions: {x+y} or ∅

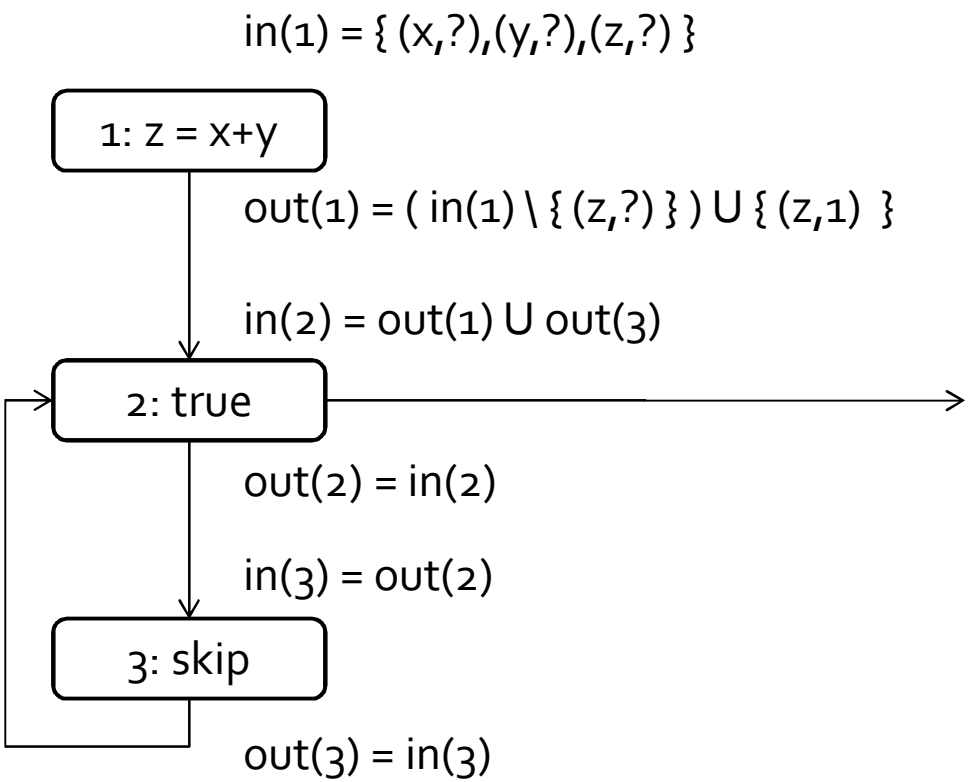
Reaching Definitions Revisited

Block	**out (lab)**
$[x := a]^{lab}$	$in(lab) \setminus \{ (x,l) \mid l \in Lab \} \cup \{ (x,lab) \}$
$[skip]^{lab}$	$in(lab)$
$[b]^{lab}$	$in(lab)$

Block	**kill**	**gen**
$[x := a]^{lab}$	$\{ (x,l) \mid l \in Lab \}$	$\{ (x,lab) \}$
$[skip]^{lab}$	$\emptyset$	$\emptyset$
$[b]^{lab}$	$\emptyset$	$\emptyset$

For each program point, which assignments <u>may</u> have been made and not overwritten, when program execution reaches this point along <u>some path</u>.

Why solution with smallest sets?



in(1) = { (x,?),(y,?),(z,?) }
in(2) = out(1) **U** out(3)
in(3) = out(2)

[z := x+y]^1;
while [true]^2 (
  [skip]^3;
)

After simplification: in(2) = in(2) U { (x,?),(y,?),(z,1) }

Many solutions: any superset of { (x,?),(y,?),(z,1) }

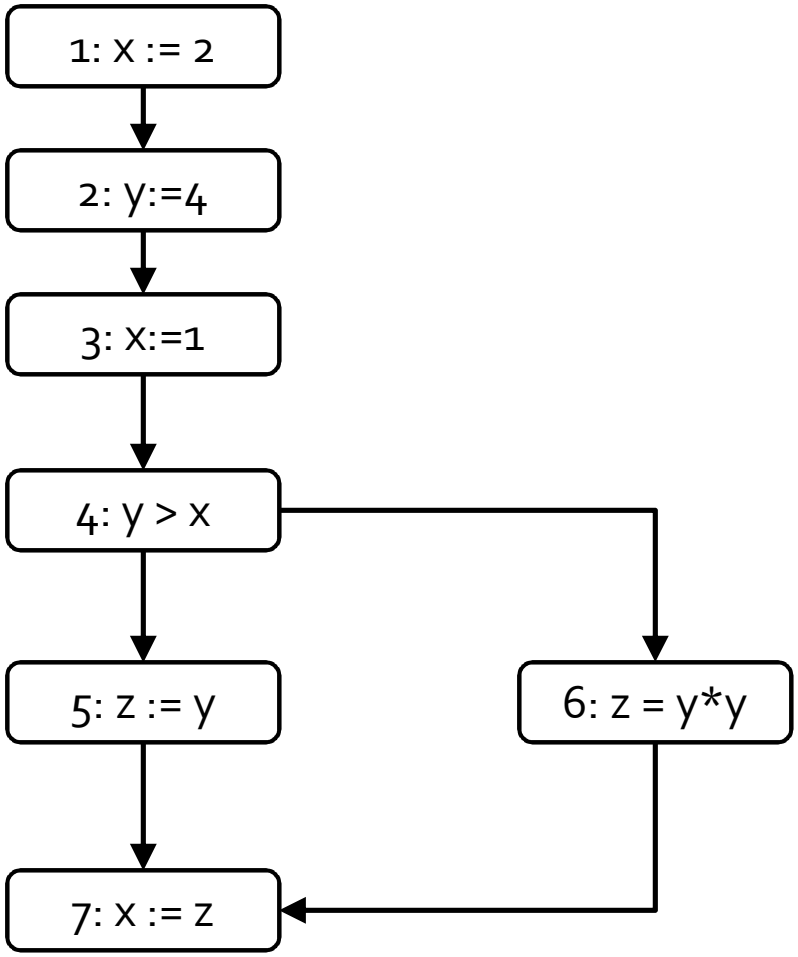
Live Variables

$[x := 2]^1;$
$[y := 4]^2;$
$[x := 1]^3;$
$(\text{if } [y > x]^4 \text{ then } [z := y]^5$
$\text{else } [z := y * y]^6);$
$[x := z]^7$

For each program point, which variables <u>may</u> be live at the exit from the point.

Live Variables

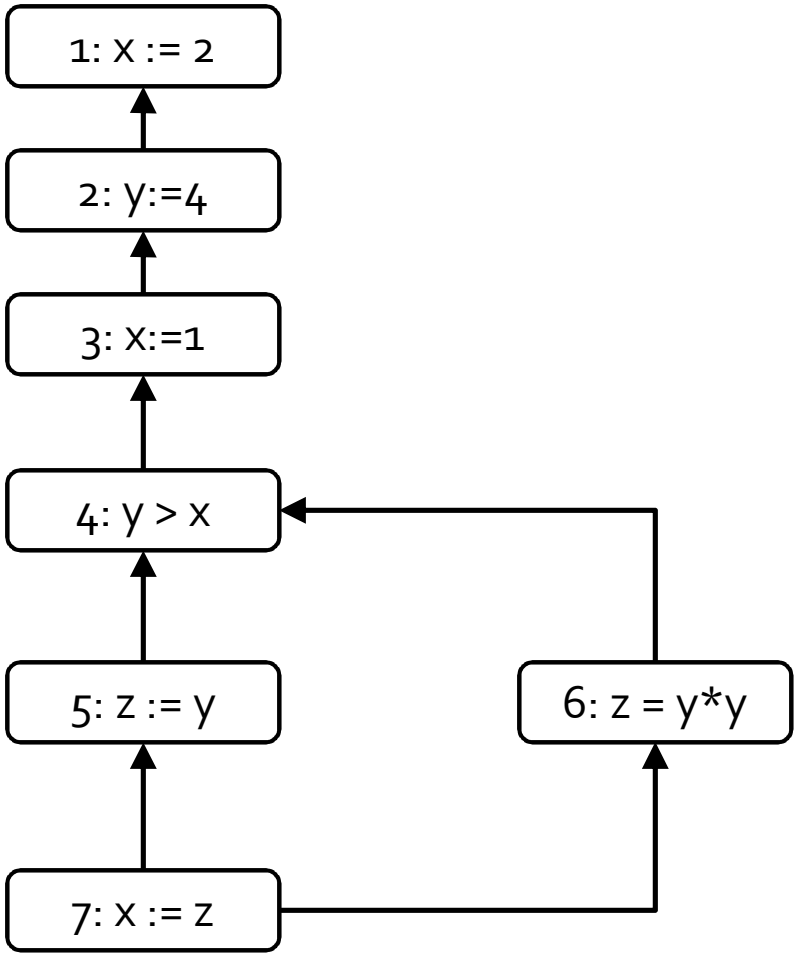
$[x := 2]^1;$
$[y := 4]^2;$
$[x := 1]^3;$
$(\text{if } [y > x]^4 \text{ then } [z := y]^5$
$\text{else } [z := y * y]^6);$
$[x := z]^7$



Live Variables

$[x := 2]^1;$
$[y := 4]^2;$
$[x := 1]^3;$
$(\text{if } [y > x]^4 \text{ then } [z := y]^5$
$\text{else } [z := y * y]^6);$
$[x := z]^7$

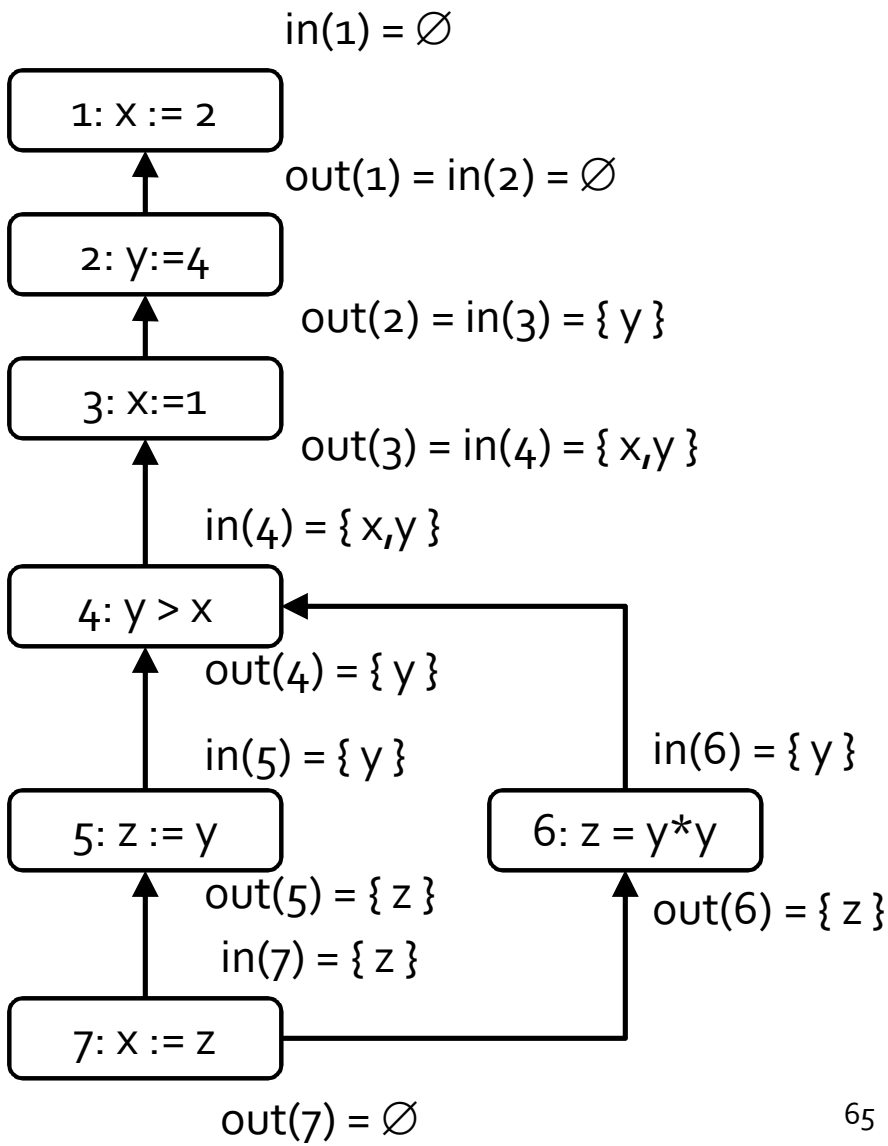
Block	kill	gen
$[x := a]^{lab}$	{ x }	{ FV(a) }
$[skip]^{lab}$	Ø	Ø
$[b]^{lab}$	Ø	FV(b)



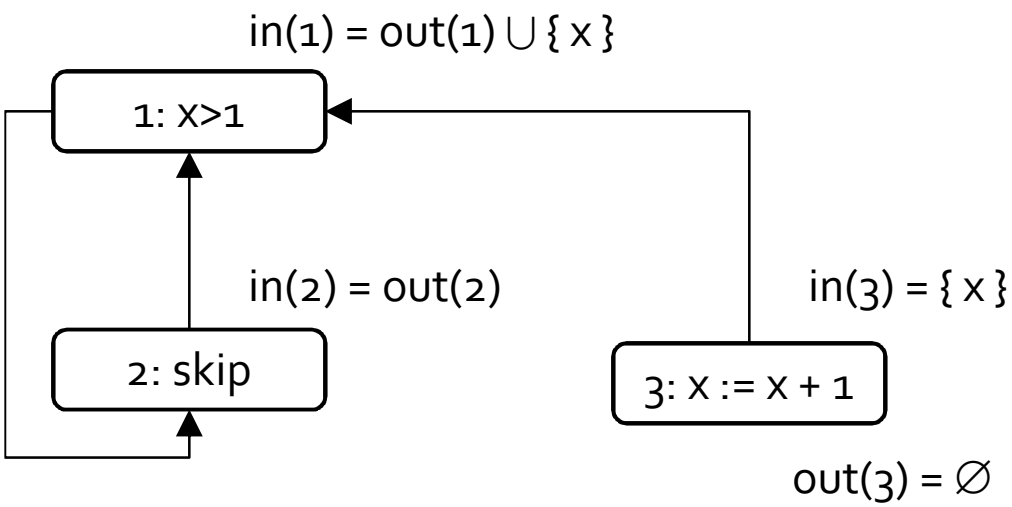
Live Variables: solution

$[x := 2]^1;$
$[y := 4]^2;$
$[x := 1]^3;$
$(\text{if } [y > x]^4 \text{ then } [z := y]^5$
$\text{else } [z := y * y]^6);$
$[x := z]^7$

Block	kill	gen
$[x := a]^{lab}$	{ x }	{ FV(a) }
$[skip]^{lab}$	Ø	Ø
$[b]^{lab}$	Ø	FV(b)



Why solution with smallest set?



out(1) = in(2) **U** in(3)
out(2) = in(1)
out(3) = ∅

while [x>1]^1 (
  [skip]^2;
)
[x := x+1]^3;

After simplification: in(1) = in(1) U { x }

Many solutions: any superset of { x }

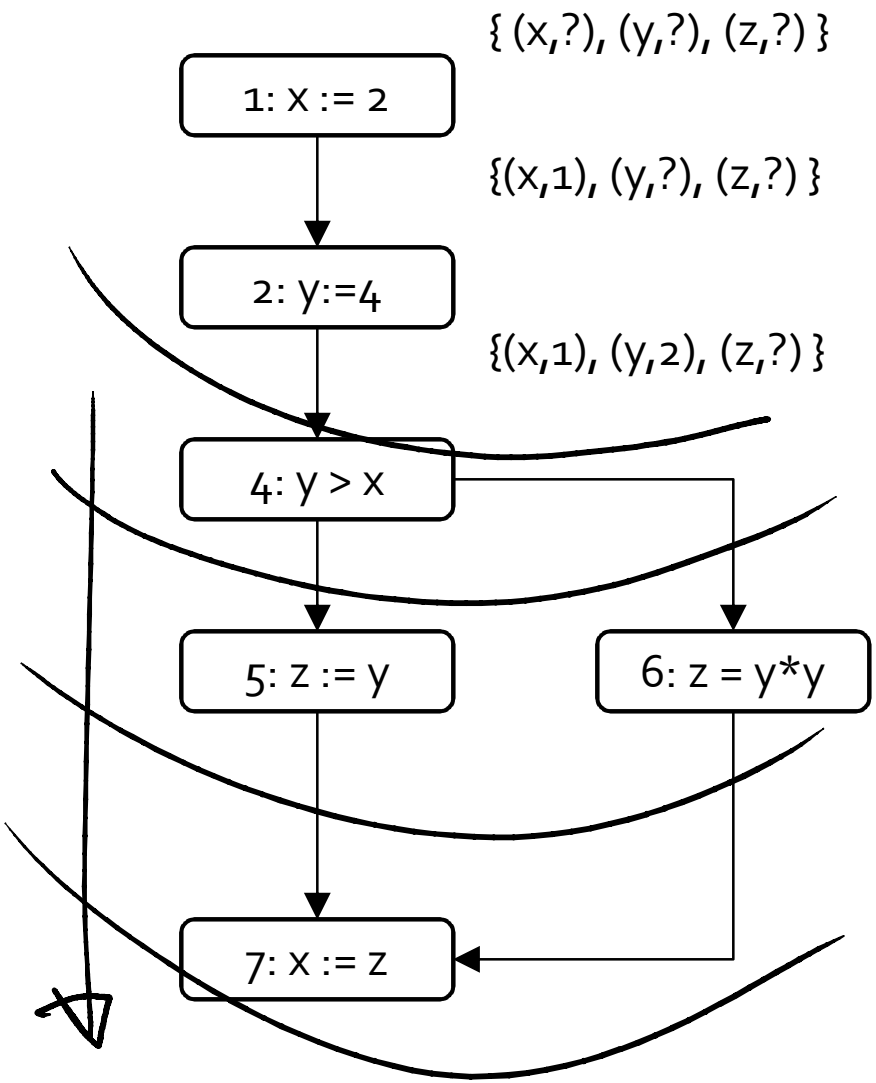
Monotone Frameworks

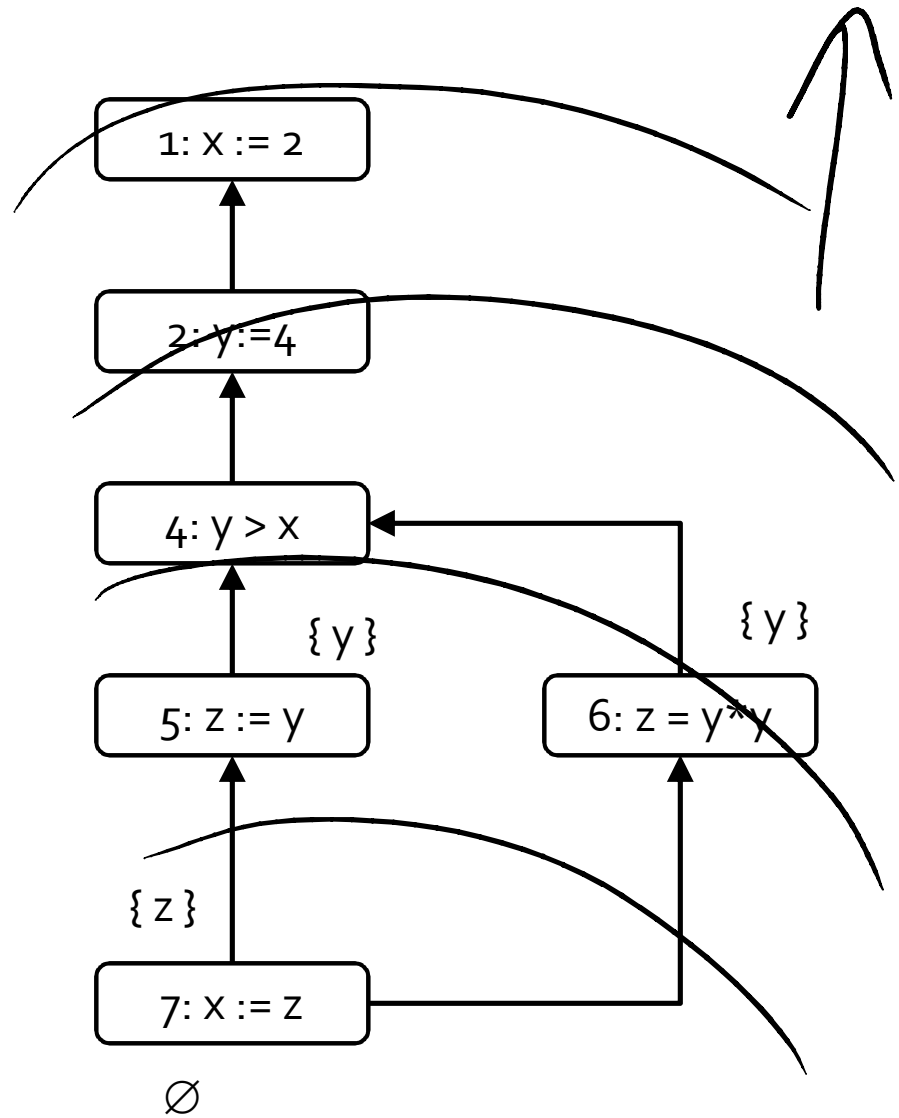
$$\text{In(lab)} = \begin{cases} \text{Initial} & \text{when lab} \in \text{Entry labels} \\ \bigsqcup \{ \text{out(lab')} \mid (\text{lab'},\text{lab}) \in \text{CFG edges} \} & \text{otherwise} \end{cases}$$

$$\text{out(lab)} = f_{\text{lab}}(\text{in(lab)})$$

- $\bigsqcup$ is $\cup$ or $\cap$
- CFG edges go either forward or backwards
- Entry labels are either initial program labels or final program labels (when going backwards)
- Initial is an initial state (or final when going backwards)
- f_{lab} is the transfer function associated with the blocks B^{lab}

Forward vs. Backward Analyses





Must vs. May Analyses

- When $\sqcup$ is $\cap$ - must analysis
  - Want largest sets the solves the equation system
  - Properties hold on all paths reaching a label (exiting a label, for backwards)

- When $\sqcup$ is $\cup$ - may analysis
  - Want smallest sets that solve the equation system
  - Properties hold at least on one path reaching a label (existing a label, for backwards)

Example: Reaching Definition

- L = $\wp$(Var×Lab) is partially ordered by $\subseteq$
- $\sqcup$ is $\cup$
- L satisfies the Ascending Chain Condition because Var × Lab is finite (for a given program)

Example: Available Expressions

- L = $\wp$(AExp) is partially ordered by $\supseteq$
- $\sqcup$ is $\cap$
- L satisfies the Ascending Chain Condition because AExp is finite (for a given program)

Analyses Summary

	Reaching Definitions	**Available Expressions**	**Live Variables**
L	$\wp$(Var x Lab)	$\wp$(AExp)	$\wp$(Var)
$\sqsubseteq$	$\subseteq$	$\supseteq$	$\subseteq$
$\sqcup$	$\cup$	$\cap$	$\cup$
$\perp$	$\emptyset$	AExp	$\emptyset$
Initial	$\{ (x,?) \mid x \in Var\}$	$\emptyset$	$\emptyset$
Entry labels	{ init }	{ init }	final
Direction	Forward	Forward	Backward
F	$\{ f: L \rightarrow L \mid \exists k,g : f(val) = (val \setminus k) \cup g \}$		
f_{lab}	$f_{lab}(val) = (val \setminus kill) \cup gen$		

Analyses as Monotone Frameworks

- Property space
  - Powerset
  - Clearly a complete lattice
- Transformers
  - Kill/gen form
  - Monotone functions (let's show it)

Monotonicity of Kill/Gen transformers

- Have to show that $x \sqsubseteq x'$ implies $f(x) \sqsubseteq f(x')$
- Assume $x \sqsubseteq x'$, then for kill set k and gen set g
  $(x \setminus k) \cup g \sqsubseteq (x' \setminus k) \cup g$

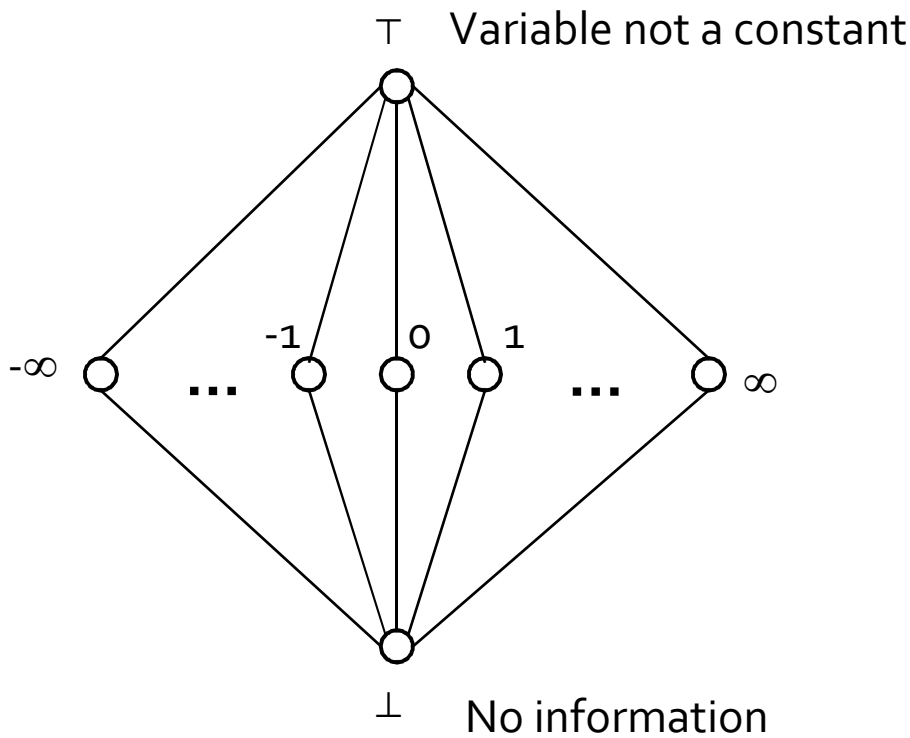
- Technically, since we want to show it for all functions in F, we also have to show that the set is closed under function composition

Distributivity of Kill/Gen transformers

- Have to show that $f(x \sqcup y) \sqsubseteq f(x) \sqcup f(y)$
- $f(x \sqcup y) = ((x \sqcup y) \setminus k) \cup g$
  $= ((x \setminus k) \sqcup (y \setminus k)) \cup g$
  $= (((x \setminus k) \cup g) \sqcup ((y \setminus k) \cup g))$
  $= f(x) \sqcup f(y)$

- Used distributivity of $\sqcup$ and $\cup$
  - Works regardless of whether $\sqcup$ is $\cup$ or $\cap$

Constant Propagation



$$\sigma \in \text{State} = (\text{Var} \rightarrow Z^{\top})_{\perp}$$

Constant Propagation

- $L = ((Var \rightarrow Z^{\top})_{\perp}, \sqsubseteq)$
- $\sigma 1 \sqsubseteq \sigma 2$ iff $\forall x: \sigma 1(x) \sqsubseteq \sigma 2(x)$
- $\sqsubseteq$ ordering in the $Z^{\top}{}_{\perp}$ lattice

- Examples:
- $[x \mapsto \perp, y \mapsto 42, z \mapsto \perp] \sqsubseteq [x \mapsto \perp, y \mapsto 42, z \mapsto 73]$
- $[x \mapsto \perp, y \mapsto 42, z \mapsto 73] \sqsubseteq [x \mapsto \perp, y \mapsto 42, z \mapsto \top]$

Constant Propagation

A: AExp $\rightarrow$ (State $\rightarrow$ Z)

$$A[\![x]\!]\sigma = \begin{cases} \bot \text{ If } \sigma = \bot \\ \sigma(x) \text{ otherwise} \end{cases} \qquad A[\![n]\!]\sigma = \begin{cases} \bot \text{ If } \sigma = \bot \\ n \text{ otherwise} \end{cases}$$

$$A[\![a1 \text{ op } a2]\!]\sigma = A[\![a1]\!]\sigma \text{ op } A[\![a2]\!]\sigma$$

Block	**out (σ)**
$[x := a]^{lab}$	$\bot$ If $\sigma = \bot$ $\sigma[x \mapsto A[\![a]\!]\sigma]$ otherwise
$[skip]^{lab}$	σ
$[b]^{lab}$	σ

Example

$[x := 42]^1;$ — $[X \mapsto \bot, y \mapsto \bot, z \mapsto \bot, w \mapsto \bot]$

$[y := 73]^2;$ — $[X \mapsto 42, y \mapsto \bot, z \mapsto \bot, w \mapsto \bot]$

(if $[?]^3$ then — $[X \mapsto 42, y \mapsto 73, z \mapsto \bot, w \mapsto \bot]$

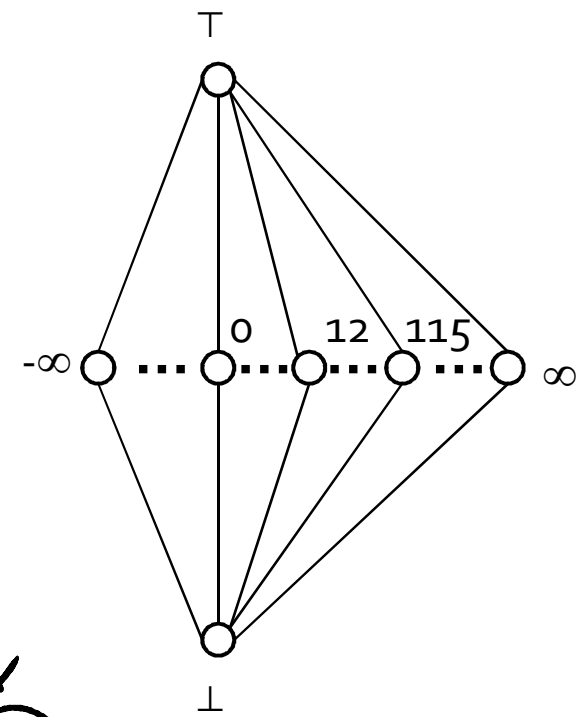
$[z := x+y]^4$ — $[X \mapsto 42, y \mapsto 73, z \mapsto \bot, w \mapsto \bot]$

else — $[X \mapsto 42, y \mapsto 73, z \mapsto 115, w \mapsto \bot]$

$[z := 12]^5$ — $[X \mapsto 42, y \mapsto 73, z \mapsto 12, w \mapsto \bot]$

$[w := z]^6$ — $[X \mapsto 42, y \mapsto 73, z \mapsto 12, w \mapsto 12]$

); — $[X \mapsto 42, y \mapsto 73, z \mapsto \top, w \mapsto 12]$



Constant Propagation is Non Distributive

- Consider the transformer f = $[\![$[y=x*x]$]\!]^{\#}$
- Consider two states $\sigma 1$, $\sigma 2$
  - $\sigma 1$(x) = 1
  - $\sigma 2$(x) = -1

$(\sigma 1 \sqcup \sigma 2)(x) = \top$	f($\sigma 1$) maps y to 1
f($\sigma 1 \sqcup \sigma 2$) maps y to $\top$	f($\sigma 2$) maps y to 1
	f($\sigma 1$) $\sqcup$ f($\sigma 2$) maps y to 1

f($\sigma 1 \sqcup \sigma 2$) $\neq$ f($\sigma 1$) $\sqcup$ f($\sigma 2$)